Big Data Mining – Comp Sci 3306
Assignment 3

Group 23

Heath Rusby a1708147
Zhao Ming Soh a1751699

Exercise 1. (Covers Q3 and Q4)

Q3. Comparison between Simple Randomised (SR) and SON

Note: The test results for this algorithm on the T40I10D100K.dat dataset is seen on the following page with the included output limited to only the larger sized itemsets for comparability.

For tests performed with the SR algorithm, sample sizes of 10% were used as this provides the best chance of the random sample accurately representing the full dataset (as explained in Q4). The support threshold for each test was optimised for each dataset. Datasets that contained many itemsets within a similar range of frequencies required a larger threshold in order to class only the itemsets that are comparatively more frequent than other itemsets within the data. My process for finding these threshold values was to steadily increase the threshold each test until I was achieving consistent results with approximately 5 – 20 frequent itemsets of the largest possible size. The selected thresholds for each dataset were as follows. A summary of the results of the SR algorithm on each dataset can be seen in the table below with only the largest frequent itemsets displayed for comparison purposes. The full outputs of each SR algorithm test can be found in "*Exercise1/outputs/simpleRandom*"

The SON algorithm I implemented splits the original dataset into chunks of maximum 4mb and performs A-Priori's algorithm on each in parallel. As a result, it considers the total dataset as opposed to the random 10% sample and is therefor an accurate representation of the frequent itemsets. The summary results of the SON algorithm tests can be seen in the table below, with full outputs available at "*Exercise1/outputs/SON/dataset/Reducer2Output /part-r-00000*" for each dataset. The results show the elimination/introduction of false negatives and positives originally presented by the SR algorithm.

Comparison of results

Dataset	Output SR Algorithm Note: the 1st number in the bracket indicates the frequency as a percentage of the sample size	SON algorithm results
T10I4D100K.dat (0.01 support thresh)	**Sample size: 0.1 (10103 of 100000)** Checking set size 1... Found 372 frequent sets. Checking set size 2... Found 15 frequent sets. Checking set size 3... Found 2 frequent sets. Completed search for itemsets. Found 389 Runtime: 5.01 seconds [33, 283, 346] (0.010096011085816094 102) [39, 704, 825] (0.010590913590022765 107) [704, 825] (0.011580718598436108 117) [217, 346] (0.01464911412451747 148) [368, 682] (0.01187766010096011 120) [390, 722] (0.010887855092546769 110) [33, 346] (0.01029397208749876 2 104) [283, 346] (0.0106898940908641 108) [789, 829] (0.012174601603484114 123) [39, 704] (0.011283777095912105 114) [33, 283] (0.010194991586657428 103) [296, 829] (0.010788874591705433 109) [368, 829] (0.01464911412451747 148) [227, 722] (0.010392952588340098 105) [39, 825] (0.011976640601801446 121) [217, 283] (0.010194991586657428 103) [227, 390] (0.011085816094229437 112)	**385 Frequent itemsets found.** [39, 704, 825] (1035) [217, 346] (1336) [227, 390] (1049) [368, 682] (1193) [368, 829] (1194) [39, 704] (1107) [39, 825] (1187) [390, 722] (1042) [704, 825] (1102) [789, 829] (1194)

T40I10D100K.dat (0.02 support thresh)	**Sample size: 0.1 (9925 of 100000)** Checking set size 1... Found 608 frequent sets. Checking set size 2... Found 1708 frequent sets. Checking set size 3... Found 6 frequent sets. Checking set size 4... Found 0 frequent sets. Completed search for itemsets. Found 2322 Runtime: 19.08 seconds [217, 368, 529] (0.0212594458438287 17 211) [368, 895, 937] (0.022267002518891688 221) [368, 529, 692] (0.023073047858942066 229) [368, 529, 829] (0.02105793450881612 209) [368, 489, 682] (0.022670025188916875 225) [198, 368, 937] (0.022468513853904283 223)	**2014 Frequent itemsets found** [217, 368, 529] (2181) [368, 489, 682] (2420) [368, 529, 692] (2373) [368, 529, 829] (2288)
Chess.dat (0.9 support thresh)	**Sample size: 0.1 (333 of 3196)** Checking set size 1... Found 13 frequent sets. Checking set size 2... Found 59 frequent sets. Checking set size 3... Found 131 frequent sets. Checking set size 4... Found 159 frequent sets. Checking set size 5... Found 107 frequent sets. Checking set size 6... Found 37 frequent sets. Checking set size 7... Found 5 frequent sets. Checking set size 8... Found 0 frequent sets. Completed search for itemsets. Found 511 Runtime: 1.27 seconds [29, 36, 40, 52, 58, 60, 62] (0.9099099099099099 303) [29, 40, 52, 56, 58, 60, 62] (0.9039039039039038 301) [29, 36, 40, 52, 56, 58, 60] (0.9009009009009009 300) [7, 29, 36, 40, 52, 58, 60] (0.9099099099099099 303) [29, 36, 40, 48, 52, 58, 60] (0.9219219219219219 307)	**622 Frequent itemsets found** [29, 36, 40, 48, 52, 58, 60] (2910) [29, 36, 40, 52, 58, 60, 62] (2878) [29, 36, 40, 52, 58, 60, 66] (2880) [7, 29, 36, 40, 52, 58, 60] (2890)
Pumsb.dat (0.95 support thresh)	**Sample size: 0.1 (4898 of 49046)** Checking set size 1... Found 13 frequent sets. Checking set size 2... Found 51 frequent sets. Checking set size 3... Found 70 frequent sets. Checking set size 4... Found 40 frequent sets. Checking set size 5... Found 9 frequent sets. Checking set size 6... Found 0 frequent sets. Completed search for itemsets. Found 183 Runtime: 1.55 seconds [170, 180, 184, 4428, 4438] (0.958554 4695) [170, 180, 184, 4428, 7062] (0.950183 4654) [170, 180, 184, 4432, 4438] (0.950183 4654) [170, 180, 184, 4434, 4438] (0.954062 4673) [170, 180, 184, 4438, 7062] (0.966721 4735) [170, 180, 184, 4438, 4940] (0.951204 4659) [180, 184, 4434, 4438, 7062] (0.95018 4654) [170, 180, 184, 4438, 7092] (0.951408 4660) [180, 184, 4428, 4438, 7062] (0.95487 4677)	**103 Frequent Itemsets found** **[170, 180, 184, 4428, 4438] (46971)** **[170, 180, 184, 4438, 7062] (47403)** **[180, 184, 4428, 4438, 7062] (46906)**
Pumsb_star.dat (0.6 support thresh)	**Sample size: 0.1 (4880 of 49046)** Checking set size 1... Found 14 frequent sets. Checking set size 2... Found 28 frequent sets. Checking set size 3... Found 45 frequent sets. Checking set size 4... Found 45 frequent sets. Checking set size 5... Found 26 frequent sets. Checking set size 6... Found 8 frequent sets. Checking set size 7... Found 1 frequent sets. Completed search for itemsets. Found 167 Runtime: 1.33 seconds [84, 161, 168, 277, 4499, 4502] (0.60881147 2971) [84, 161, 168, 4493, 4496, 4502] (0.6258196 3054) [84, 168, 4493, 4496, 4499, 4502] (0.624385 3047) [161, 168, 4493, 4496, 4499, 4502] (0.62438 3047) [84, 161, 168, 4496, 4499, 4502] (0.6565573 3204) [84, 161, 168, 4493, 4496, 4499] (0.6346311 3097) [84, 161, 4493, 4496, 4499, 4502] (0.625409 3052) [84, 161, 168, 4493, 4499, 4502] (0.6243852 3047) [84, 161, 168, 4493, 4496, 4499, 4502] (0.62438 3047)	**165 Frequent Itemsets found** [84, 161, 168, 4493, 4496, 4499, 4502] (30445) [161, 168, 4493, 4496, 4499, 4502] (30445) [84, 161, 168, 277, 4499, 4502] (30204) [84, 161, 168, 4493, 4496, 4499, 4502] (30445) [84, 161, 168, 4493, 4496, 4499] (30898) [84, 161, 168, 4493, 4496, 4502] (30495) [84, 161, 168, 4493, 4499, 4502] (30445) [84, 161, 168, 4496, 4499, 4502] (32212) [84, 161, 4493, 4496, 4499, 4502] (30490) [84, 168, 4493, 4496, 4499, 4502] (30445)

The key difficulty with the implementation of the SON algorithm was to overwrite mapreduces default design to run one map thread on each line of the input file as opposed

to one map for each file. I navigated around this by keeping a “register” file that contained the list of split filenames and then passed this file into the first mapreduce job. The first job then creates a mapper to handle each filename on each line.

Q4. Sample Size Comparison

The results of testing with variable sample rates indicate that the use of a smaller sample (i.e. 1-2% of the original dataset) yielded results that fluctuate greatly with each run. This is likely due to vast differences in the data used to fill each small sample. As the sample size increased to 5-10% the results begin to show consistency as the larger samples are able to better represent the dataset as a whole. This can be seen in the results below where, as the sample size increased, the frequent itemsets converged towards those found by the SON algorithm after considering the whole dataset. (further supporting evidence from tests with other dataset can be found in the appendix)

The tendency to produce significant false positives and negatives at low sample sizes is dependent upon the dataset being considered. Datasets with larger variation in itemsets and frequency will require a larger sample size to capture all important frequent itemsets.

Test Results – Simple Randomized Algorithm - T40I10D100K.dat

Sample size: 0.01 (980 of 100000) Checking set size 1... Found 598 frequent sets. Checking set size 2... Found 2212 frequent sets. Checking set size 3... Found 195 frequent sets. Checking set size 4... Found 194 frequent sets. Checking set size 5... Found 160 frequent sets. Checking set size 6... Found 91 frequent sets. Checking set size 7... Found 36 frequent sets. Checking set size 8... Found 9 frequent sets. Checking set size 9... Found 1 frequent sets. Completed search for itemsets. Found 3496 Runtime: 3.03 seconds Frequent itemsets (Displaying all sets with size >= 8): [177, 275, 298, 375, 489, 710, 745, 934] (0.02040816326530612 20) [275, 298, 375, 489, 579, 710, 745, 934] (0.02040816326530612 20) [177, 275, 298, 375, 489, 579, 710, 745] (0.02040816326530612 20) [177, 298, 375, 489, 579, 710, 745, 934] (0.02040816326530612 20) [177, 275, 298, 375, 489, 579, 745, 934] (0.02040816326530612 20) [177, 275, 298, 375, 489, 579, 710, 934] (0.02040816326530612 20) [177, 275, 298, 375, 579, 710, 745, 934] (0.02040816326530612 20) [177, 275, 298, 489, 579, 710, 745, 934] (0.02040816326530612 20) [177, 275, 375, 489, 579, 710, 745, 934] (0.02040816326530612 20) [177, 275, 298, 375, 489, 579, 710, 745, 934] (0.02040816326530612 20)	**Sample size: 0.02 (2106 of 100000)** Checking set size 1... Found 603 frequent sets. Checking set size 2... Found 1818 frequent sets. Checking set size 3... Found 60 frequent sets. Checking set size 4... Found 17 frequent sets. Checking set size 5... Found 2 frequent sets. Completed search for itemsets. Found 2500 Runtime: 4.59 seconds Frequent itemsets (Displaying all sets with size >= 4): [93, 480, 674, 712] (0.020892687559354226 44) [480, 484, 674, 712] (0.020417853751187084 43) [220, 484, 674, 712] (0.020892687559354226 44) [413, 538, 826, 956] (0.020417853751187084 43) [220, 480, 674, 712] (0.020892687559354226 44) [93, 220, 674, 712] (0.021367521367521368 45) [220, 484, 674, 759] (0.020892687559354226 44) [93, 484, 712, 759] (0.020417853751187084 43) [93, 220, 712, 759] (0.020417853751187084 43) [93, 220, 484, 674] (0.020417853751187084 43) [93, 220, 484, 712] (0.020892687559354226 44) [93, 484, 674, 712] (0.020892687559354226 44) [93, 220, 712, 854] (0.020417853751187084 43) [93, 220, 480, 712] (0.020417853751187084 43) [93, 220, 480, 674] (0.020417853751187084 43) [93, 674, 712, 759] (0.020417853751187084 43) [220, 674, 712, 854] (0.020417853751187084 43) [93, 220, 484, 674, 712] (0.020417853751187084 43) [93, 220, 480, 674, 712] (0.020417853751187084 43)
Sample size: 0.05 (5098 of 100000) Checking set size 1... Found 615 frequent sets. Checking set size 2... Found 1818 frequent sets. Checking set size 3... Found 9 frequent sets. Checking set size 4... Found 0 frequent sets. Completed search for itemsets. Found 2442 Runtime: 9.74 seconds Frequent itemsets (Displaying all sets with size >= 3): [368, 692, 829] (0.02020400156924284 103) [217, 368, 529] (0.02157708905453119 110) [368, 529, 692] (0.02785406041584935 142) [368, 675, 682] (0.02000784621420165 102) [368, 529, 829] (0.02157708905453119 110) [205, 509, 966] (0.020400156924284034 104) [368, 489, 682] (0.02314633189486073 118) [339, 598, 966] (0.020400156924284034 104) [198, 368, 937] (0.024127108670066694 123)	**Sample size: 0.1 (9925 of 100000)** Checking set size 1... Found 608 frequent sets. Checking set size 2... Found 1708 frequent sets. Checking set size 3... Found 6 frequent sets. Checking set size 4... Found 0 frequent sets. Completed search for itemsets. Found 2322 Runtime: 19.08 seconds Frequent itemsets (Displaying on sets with size >= 3): [217, 368, 529] (0.0212594458438287 17 211) [368, 895, 937] (0.022267002518891688 221) [368, 529, 692] (0.023073047858942066 229) [368, 529, 829] (0.02105793450881612 209) [368, 489, 682] (0.022670025188916875 225) [198, 368, 937] (0.022468513853904283 223)

APPENDIX – Additional Testing Data

Sample Size Test Results – Simple Randomized Algorithm - pumsb.dat

Sample size: 0.01 (453 of 49046)
Checking set size 1... Found 14 frequent sets.
Checking set size 2... Found 57 frequent sets.
Checking set size 3... Found 89 frequent sets.
Checking set size 4... Found 66 frequent sets.
Checking set size 5... Found 25 frequent sets.
Checking set size 6... Found 4 frequent sets.
Checking set size 7... Found 0 frequent sets.
Completed search for itemsets. Found 255
Runtime: 1.26 seconds

[170, 180, 184, 4438, 4940, 7062] (0.9514348785871964 431)
[180, 184, 4426, 4428, 4438, 7062] (0.9558498896247241 433)
[170, 180, 184, 4426, 4428, 4438] (0.9558498896247241 433)
[170, 180, 184, 4428, 4438, 7062] (0.9602649006622517 435)

Sample size: 0.02 (1004 of 49046)
Checking set size 1... Found 15 frequent sets.
Checking set size 2... Found 49 frequent sets.
Checking set size 3... Found 58 frequent sets.
Checking set size 4... Found 27 frequent sets.
Checking set size 5... Found 4 frequent sets.
Checking set size 6... Found 0 frequent sets.
Completed search for itemsets. Found 153
Runtime: 1.28 seconds

[180, 184, 4434, 4438, 7062] (0.952191235059761 956)
[170, 180, 184, 4432, 4438] (0.951195219123506 955)
[180, 184, 4432, 4438, 7062] (0.9591633466135459 963)
[170, 180, 184, 4438, 7062] (0.9641434262948207 968)

Sample size: 0.05 (2350 of 49046)
Checking set size 1... Found 14 frequent sets.
Checking set size 2... Found 61 frequent sets.
Checking set size 3... Found 91 frequent sets.
Checking set size 4... Found 67 frequent sets.
Checking set size 5... Found 21 frequent sets.
Checking set size 6... Found 2 frequent sets.
Checking set size 7... Found 0 frequent sets.
Completed search for itemsets. Found 256
Runtime: 1.41 seconds

[180, 184, 4426, 4428, 4438] (0.9506382978723404 2234)
[180, 184, 4432, 4438, 7062] (0.9514893617021276 2236)
[170, 180, 184, 7062, 7092] (0.9506382978723404 2234)
[180, 184, 4428, 4438, 4940] (0.9502127659574469 2233)
[170, 180, 184, 4438, 7092] (0.9582978723404255 2252)
[170, 184, 4438, 4940, 7062] (0.9553191489361702 2245)
[170, 180, 4438, 4940, 7062] (0.9553191489361702 2245)
[170, 184, 4428, 4438, 7062] (0.9527659574468085 2239)
[170, 180, 4428, 4438, 7062] (0.9527659574468085 2239)
[170, 180, 184, 4940, 7062] (0.9574468085106383 2250)
[180, 184, 4438, 4940, 7062] (0.9608510638297872 2258)
[180, 184, 4428, 4438, 7062] (0.9582978723404255 2252)
[180, 184, 4426, 4438, 7062] (0.9502127659574469 2233)
[170, 180, 184, 4426, 4438] (0.9540425531914893 2242)
[170, 180, 184, 4428, 4438] (0.9621276595744681 2261)
[170, 180, 184, 4428, 7062] (0.9548936170212766 2244)
[170, 180, 184, 4432, 4438] (0.9548936170212766 2244)
[180, 184, 4438, 7062, 7092] (0.9540425531914893 2242)
[170, 180, 184, 4434, 4438] (0.9531914893617022 2240)
[170, 180, 184, 4438, 7062] (0.9727659574468085 2286)
[170, 180, 184, 4438, 4940] (0.9651063829787234 2268)
[170, 180, 184, 4438, 4940, 7062] (0.9553191489361702 2245)
[170, 180, 184, 4428, 4438, 7062] (0.9527659574468085 2239)

Sample size: 0.1 (4898 of 49046)
Checking set size 1... Found 13 frequent sets.
Checking set size 2... Found 51 frequent sets.
Checking set size 3... Found 70 frequent sets.
Checking set size 4... Found 40 frequent sets.
Checking set size 5... Found 9 frequent sets.
Checking set size 6... Found 0 frequent sets.
Completed search for itemsets. Found 183
Runtime: 1.55 seconds

[170, 180, 184, 4428, 4438] (0.9585545120457329 4695)
[170, 180, 184, 4428, 7062] (0.9501837484687627 4654)
[170, 180, 184, 4432, 4438] (0.9501837484687627 4654)
[170, 180, 184, 4434, 4438] (0.95406288280931 4673)
[170, 180, 184, 4438, 7062] (0.9667211106574112 4735)
[170, 180, 184, 4438, 4940] (0.9512045732952226 4659)
[180, 184, 4434, 4438, 7062] (0.9501837484687627 4654)
[170, 180, 184, 4438, 7092] (0.9514087382605145 4660)
[180, 184, 4428, 4438, 7062] (0.9548795426704777 4677)

Sample Size Test Results – Simple Randomized Algorithm - chess.dat

Sample size: 0.01 (41 of 3196)

Checking set size 1... Found 13 frequent sets.
Checking set size 2... Found 76 frequent sets.
Checking set size 3... Found 262 frequent sets.
Checking set size 4... Found 589 frequent sets.
Checking set size 5... Found 903 frequent sets.
Checking set size 6... Found 959 frequent sets.
Checking set size 7... Found 701 frequent sets.
Checking set size 8... Found 342 frequent sets.
Checking set size 9... Found 104 frequent sets.
Checking set size 10... Found 17 frequent sets.
Checking set size 11... Found 1 frequent sets.
Completed search for itemsets. Found 3967
Runtime: 1.32 seconds

[25, 29, 34, 36, 40, 52, 56, 58, 60, 62] (0.926829268292683 38)
[25, 29, 34, 36, 40, 48, 52, 56, 58, 62] (0.9024390243902439 37)
[25, 29, 34, 40, 48, 52, 56, 58, 60, 62] (0.9024390243902439 37)
[25, 29, 34, 36, 40, 48, 52, 58, 60, 62] (0.926829268292683 38)
[9, 25, 29, 34, 40, 52, 56, 58, 60, 62] (0.9024390243902439 37)
[25, 29, 36, 40, 48, 52, 56, 58, 60, 62] (0.9024390243902439 37)
[29, 34, 36, 40, 48, 52, 56, 58, 60, 62] (0.9024390243902439 37)
[25, 29, 34, 36, 40, 48, 52, 56, 60, 62] (0.9024390243902439 37)
[9, 25, 29, 34, 36, 40, 52, 56, 58, 60] (0.9024390243902439 37)
[25, 29, 34, 36, 40, 48, 52, 56, 58, 60] (0.926829268292683 38)
[9, 25, 29, 34, 36, 40, 52, 58, 60, 62] (0.9024390243902439 37)
[25, 34, 36, 40, 48, 52, 56, 58, 60, 62] (0.9024390243902439 37)
[9, 25, 29, 34, 36, 40, 48, 52, 58, 60] (0.9024390243902439 37)
[7, 25, 29, 34, 40, 52, 56, 58, 60, 62] (0.9024390243902439 37)
[7, 25, 29, 34, 36, 40, 52, 56, 58, 60] (0.9024390243902439 37)
[25, 29, 34, 36, 48, 52, 56, 58, 60, 62] (0.9024390243902439 37)
[25, 29, 34, 36, 40, 48, 56, 58, 60, 62] (0.9024390243902439 37)
[25, 29, 34, 36, 40, 48, 52, 56, 58, 60, 62] (0.9024390243902439 37)

Sample size: 0.02 (61 of 3196)

Checking set size 1... Found 13 frequent sets.
Checking set size 2... Found 51 frequent sets.
Checking set size 3... Found 94 frequent sets.
Checking set size 4... Found 93 frequent sets.
Checking set size 5... Found 50 frequent sets.
Checking set size 6... Found 13 frequent sets.
Checking set size 7... Found 1 frequent sets.
Completed search for itemsets. Found 315
Runtime: 1.07 seconds

[29, 36, 40, 52, 58, 60] (0.9180327868852459 56)
[36, 40, 48, 52, 58, 62] (0.9016393442622951 55)
[29, 34, 36, 40, 52, 58] (0.9016393442622951 55)
[29, 36, 52, 58, 60, 62] (0.9016393442622951 55)
[29, 40, 52, 58, 60, 62] (0.9016393442622951 55)
[29, 36, 48, 52, 58, 60] (0.9180327868852459 56)
[29, 36, 40, 52, 58, 62] (0.9180327868852459 56)
[29, 40, 48, 52, 58, 60] (0.9016393442622951 55)
[29, 36, 40, 48, 52, 60] (0.9016393442622951 55)
[29, 36, 40, 48, 52, 58] (0.9180327868852459 56)
[29, 36, 40, 48, 58, 60] (0.9016393442622951 55)
[29, 36, 48, 52, 58, 62] (0.9016393442622951 55)
[36, 40, 48, 52, 58, 60] (0.9016393442622951 55)
[29, 36, 40, 48, 52, 58, 60] (0.9016393442622951 55)

Sample size: 0.05 (146 of 3196)

Checking set size 1... Found 12 frequent sets.
Checking set size 2... Found 61 frequent sets.
Checking set size 3... Found 150 frequent sets.
Checking set size 4... Found 205 frequent sets.
Checking set size 5... Found 163 frequent sets.
Checking set size 6... Found 73 frequent sets.
Checking set size 7... Found 16 frequent sets.
Checking set size 8... Found 1 frequent sets.
Completed search for itemsets. Found 681
Runtime: 1.13 seconds

[29, 36, 40, 52, 58, 60, 66] (0.9315068493150684 136)
[29, 36, 40, 52, 58, 60, 62] (0.9178082191780822 134)
[29, 36, 40, 48, 52, 58, 66] (0.9041095890410958 132)
[29, 40, 52, 58, 60, 62, 66] (0.9178082191780822 134)
[7, 29, 40, 52, 58, 60, 62] (0.9041095890410958 132)
[36, 40, 52, 58, 60, 62, 66] (0.9041095890410958 132)
[7, 29, 36, 40, 58, 60, 66] (0.9041095890410958 132)
[7, 29, 36, 40, 52, 58, 60] (0.9178082191780822 134)
[36, 40, 48, 52, 58, 60, 66] (0.910958904109589 133)
[29, 36, 40, 48, 58, 60, 66] (0.910958904109589 133)
[29, 36, 40, 48, 52, 60, 66] (0.9041095890410958 132)
[7, 29, 40, 52, 58, 60, 66] (0.9178082191780822 134)
[29, 40, 48, 52, 58, 60, 66] (0.9041095890410958 132)
[29, 36, 48, 52, 58, 60, 66] (0.9041095890410958 132)
[29, 36, 40, 58, 60, 62, 66] (0.9041095890410958 132)
[29, 36, 40, 48, 52, 58, 60] (0.9246575342465754 135)
[29, 36, 40, 48, 52, 58, 60, 66] (0.9041095890410958 132)

Sample size: 0.1 (333 of 3196)

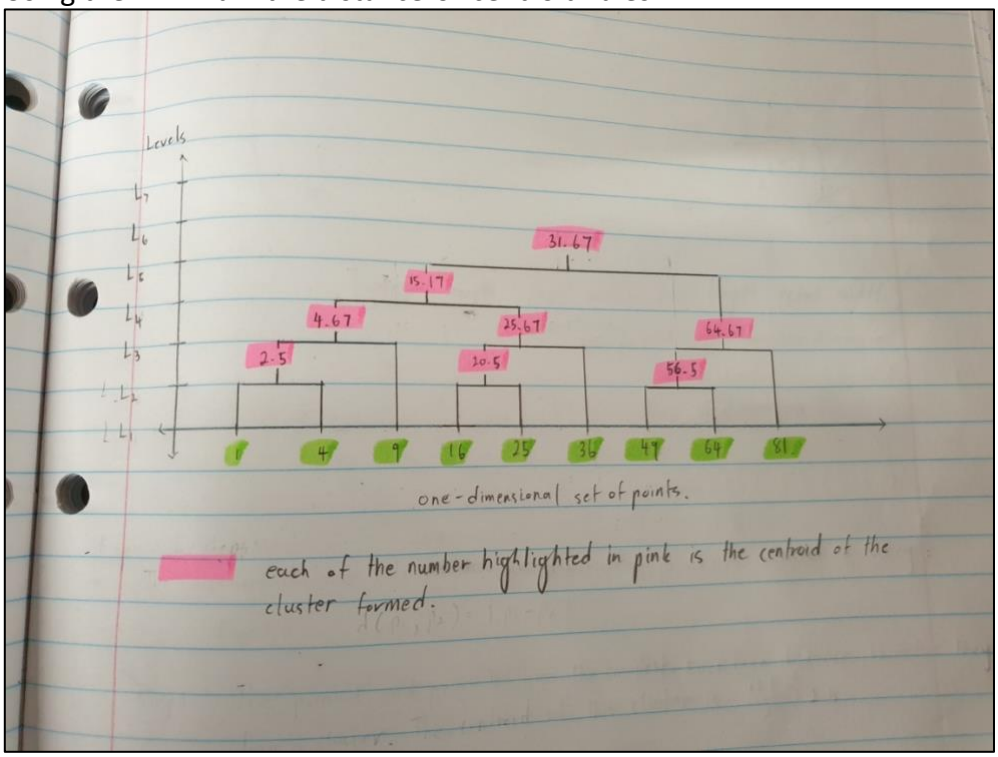
Checking set size 1... Found 13 frequent sets.
Checking set size 2... Found 59 frequent sets.
Checking set size 3... Found 131 frequent sets.
Checking set size 4... Found 159 frequent sets.
Checking set size 5... Found 107 frequent sets.
Checking set size 6... Found 37 frequent sets.
Checking set size 7... Found 5 frequent sets.
Checking set size 8... Found 0 frequent sets.
Completed search for itemsets. Found 511
Runtime: 1.27 seconds

[29, 36, 40, 52, 58, 60, 62] (0.9099099099099099 303)
[29, 40, 52, 56, 58, 60, 62] (0.9039039039039038 301)
[29, 36, 40, 52, 56, 58, 60] (0.9009009009009009 300)
[7, 29, 36, 40, 52, 58, 60] (0.9099099099099099 303)
[29, 36, 40, 48, 52, 58, 60] (0.9219219219219219 307)

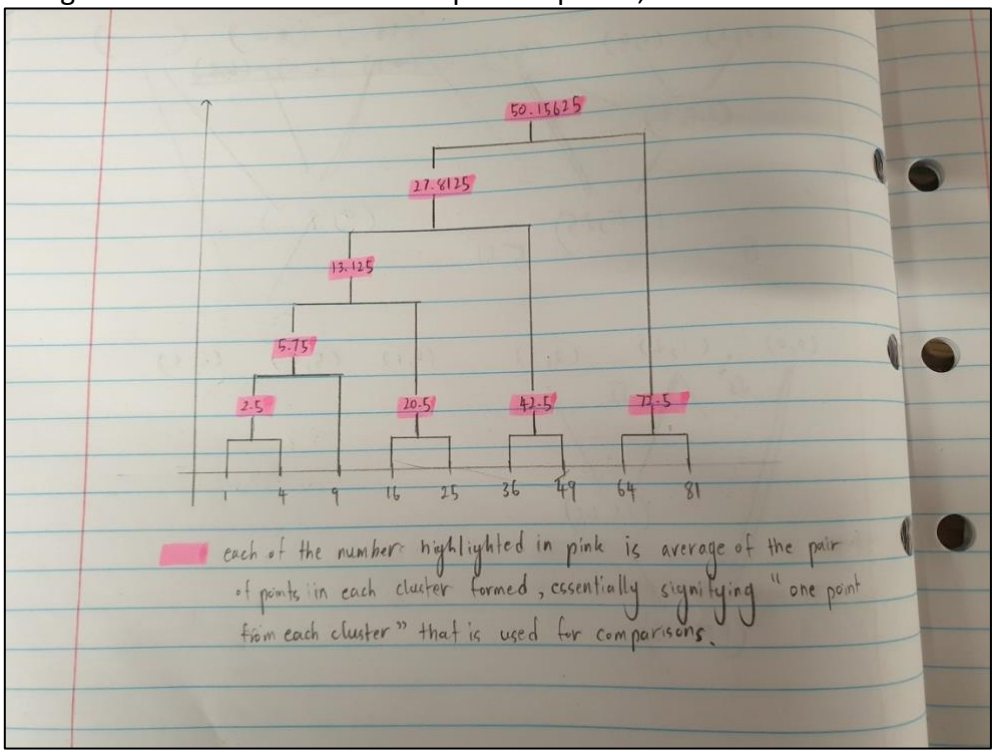
Exercise 2:

Question 1:

a. Using the minimum the distance-of-centroid rules:



b. Using the minimum distance of all pairs of points, one from each cluster:



Question 2:

a. Comparing the before and after K-means clustering plots of the first 2 dimensions of the iris dataset:

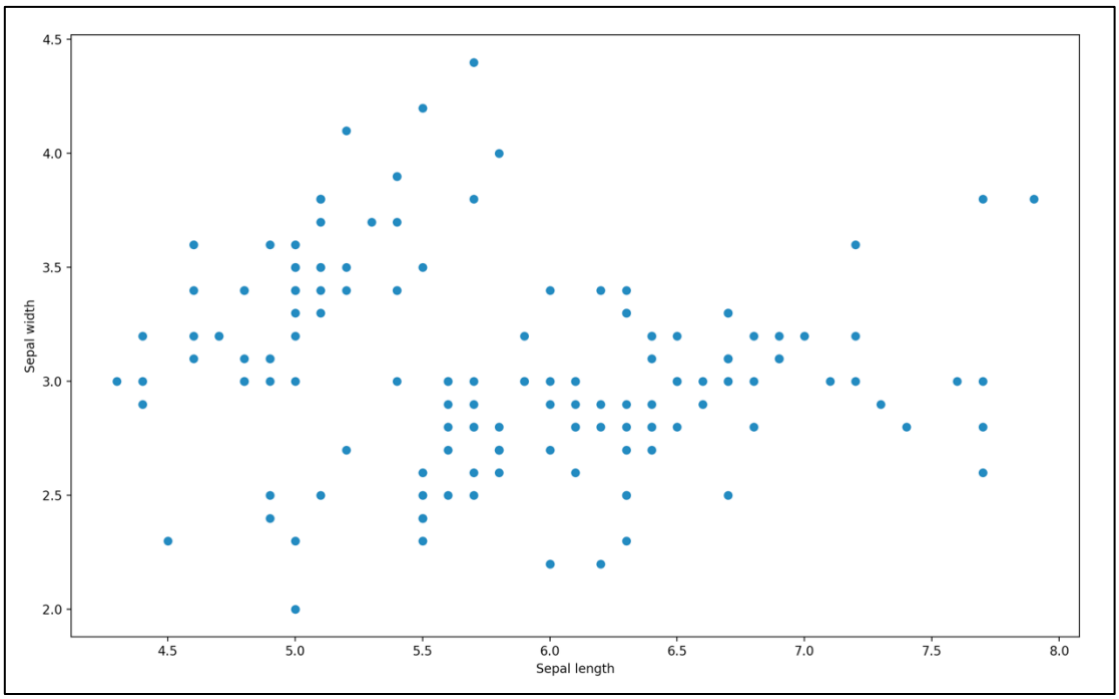


Figure 1: 2-dimensional plot of the first 2 dimensions of the iris dataset where x-axis is the sepal length and y-axis is the sepal witdh before K-means Clustering is applied.

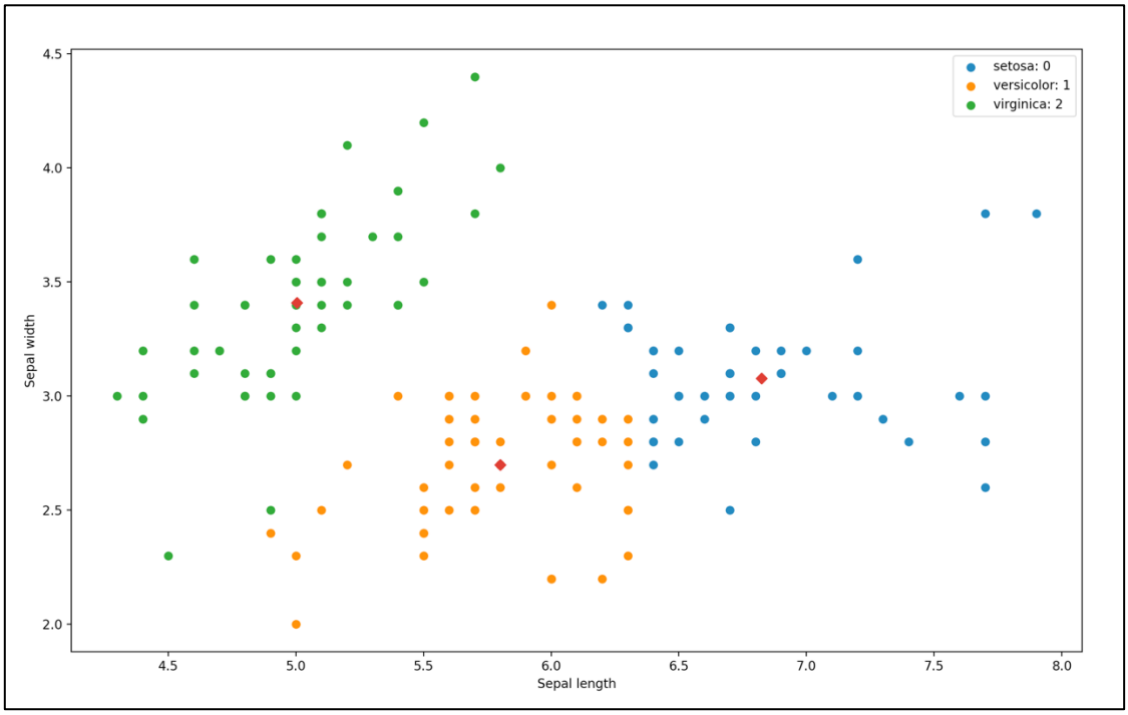


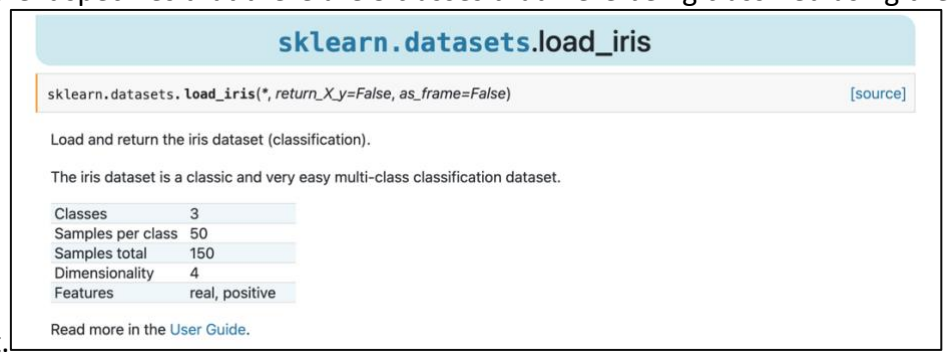
Figure 2: 2-dimensional plot of the first 2 dimensions of the iris dataset where x-axis is the sepal length and y-axis is the sepal witdh after K-means Clustering is applied. (Converged K-means)

- The red diamonds are the centroids of each cluster.
- **Description about the number of rounds taken for the centroids to converged:**

- In the code, I implemented a while loop that is conditioned to the "treshold of the difference between the old centroid in previous round and the new centroid in the new round", once the treshold is breached, the centroids will converge with an arbitrary number of rounds due to the nature of the random intialisation of the first centroids.
- Example of the centroids in each round:

```
Number of Rounds:  6
Centroid 1                  Centroid 2                  Centroid 3
-------------------------   -------------------------   -------------------------
[4.4 3. ]                   [6.2 3.4]                   [5.7 2.8]
[4.84210526 3.24736842]     [6.54098361 3.22131148]     [5.75490196 2.71960784]
[4.98627451 3.37843137]     [6.75490196 3.09019608]     [5.78541667 2.68125    ]
[5.00392157 3.40980392]     [6.81276596 3.07446809]     [5.79038462 2.69615385]
[5.006 3.428]               [6.81276596 3.07446809]     [5.77358491 2.69245283]
[5.006 3.428]               [6.81276596 3.07446809]     [5.77358491 2.69245283]
```

b. The k value or better known as the number of clusters that I've picked is 3 due to prior knowledge of the iris dataset. This prior is obtained from scikit-learn datasets library where it specifies that there are 3 classes that were being classified using the iris dataset.



sklearn.datasets.load_iris

`sklearn.datasets.load_iris(*, return_X_y=False, as_frame=False)` [source]

Load and return the iris dataset (classification).

The iris dataset is a classic and very easy multi-class classification dataset.

Classes	3
Samples per class	50
Samples total	150
Dimensionality	4
Features	real, positive

Read more in the User Guide.

   a. Class 0 is the classification for iris setosa.
   b. Class 1 is the classification for iris versicolor.
   c. Class 2 is the classification for iris virginica.

Exercise 3:

1. The worst-case that can happen for the 3 advertisers A,B and C on the query stream of xxyyzz for the greedy algorithm are as follows:
   a. BBCC_ _
   b. CCBB_ _
   c. BCBC_ _
   d. CBCB_ _
   e. CBBC_ _
   f. BCCB_ _

   By exhausting the budget of the advertiser B or C first will guarantee that the greedy algorithm will assign at least 4 out of the 6 queries. Therefore, the worst-case of the greedy algorithm is proven.

2. Another sequence of queries that will guarantee that the greedy algorithm will assign as few as half the queries of an optimal algorithm would be as follow:
   a. xxzz or yyzz

i. If advertiser C were to be assigned as the first 2 advertisers for the sequence, then advertisers B and A will not be selected for the rest of the queries because query z is not bid by both of them.
    1. CC_ _