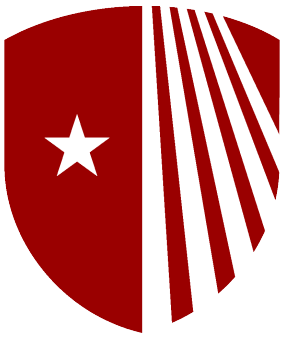
Star-Cubing Algorithm: Final Version

ESE 589

Zahin Huq & Hualin Zheng

October 2019



Stony Brook University

Contents

1 Introduction

Star-Cubing is a method of computing iceberg data cubes developed by Xin et. al.[1] Utilizing the advantages of the top down and bottom up approach, its goal is to limit the amount of interactions between the external memory and the processor. This is done by computing cubes with aggregations on multiple dimensions simultaneously and pruning data based on iceberg conditions for the attributes of a given data set. Feng et. al. describes the process of the Star-Cubing method in their study of range cubing.[2]

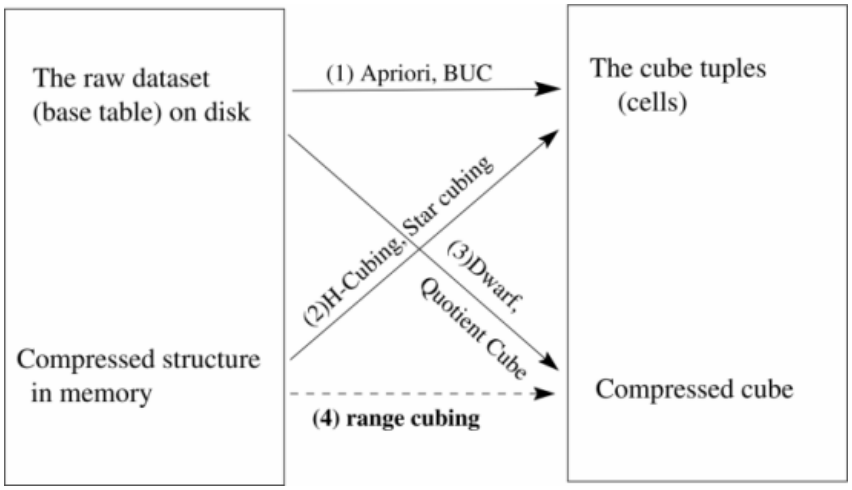


Figure 1: From figure 1 of "Range cube: efficient cube computation by exploiting data correlation." This figure demonstrates the differences between different cubing methods.

Star-Cubing involves organizing a data set into a tree structure based on its dimensions, where the variables are sorted into star variables if they do not meet a certain iceberg threshold. Utilizing simultaneous aggregation such as in Multiway aggregation and pruning with the Apriori condition such as in BUC, Star-Cubing provides a more efficient way to compute iceberg data cubes. In this paper, we analyze the efficiency of Star-Cubing and its implementation with various benchmarks.

2 Algorithm

The Star-Cubing Algorithm can be thought of as consisting of several different steps.

1. Organize data into a star table by comparing tuple elements to a threshold value and generate the star tree with lexicographical ordering.

2. Top down, depth first traversal and based on Apriori condition using the minimum support value while keeping count at each node.

3. Bottom up back traversal at the current tree, print all paths in the tree in reverse order of generation and add nodes to left subtrees until reaching a sibling, then return to step 2.

Figure 2 shows the flow chart of the algorithm in adherence to the code developed and the pseudocode generated in "Data Mining: Concepts and Techniques" by J. Han.[3]

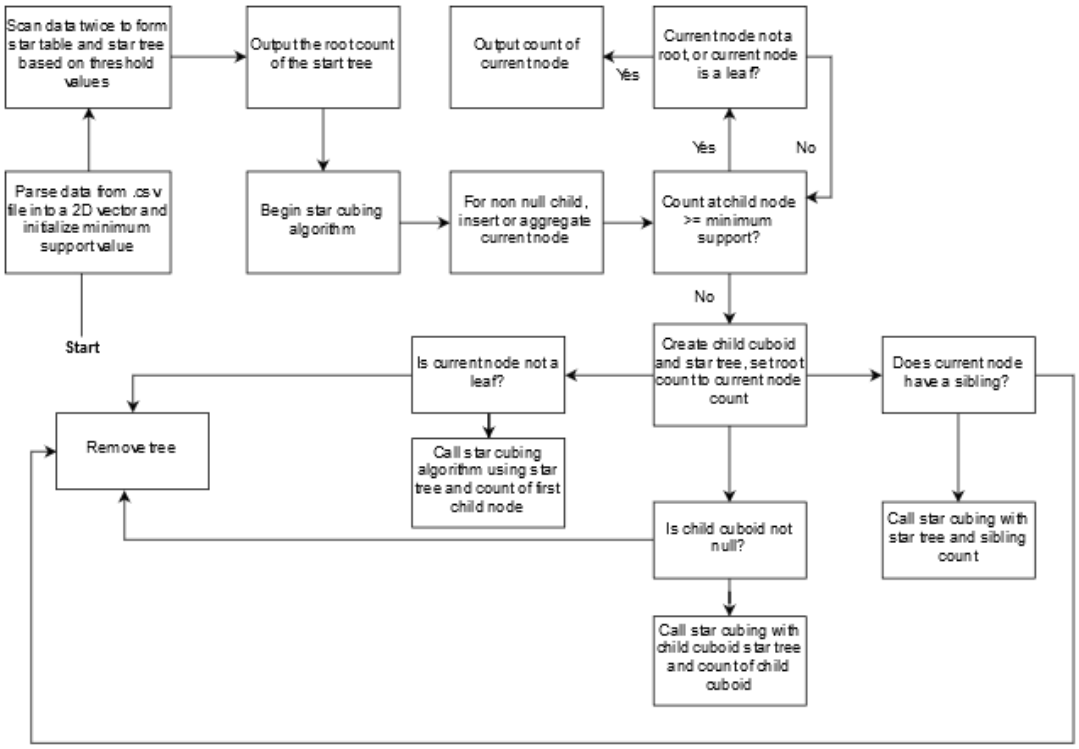


Figure 2: The flow chart of the Star-Cubing Algorithm. The initial input is the relational table (the data) and the minimum support threshold used for pruning. The output are all iceberg (or full data cubes), the star tree, and the star table.

It is important to note the case by case variability of the Star-Cubing Algorithm. While the Star-Cubing Algorithm described uses a global iceberg condition and uses count() as the iceberg condition, the algorithm can be made to incorporate dimension dependent thresholds that are different from count() i.e. average() or sum().

3 Software Implementation

We use C++ to implement Star-Cubing Algorithm. The "time.h" to check the execution time. The "map" to replicate associative arrays. The "lambda.hpp" to get special math function value. All these special functions and header need extra library files. We use Boost 1.71 and Cygwin compiler's library. We use Cygwin because it has all the special header files we needed.

The function structure for our implementation are:

Function	Input	Output
cuboid function	**data value and ice-berg**	**cuboid array**
star table function	**ice-berg**	**star table**
star tree function	**data value and root**	**star tree**
star cubing function	**star tree, root and cuboid**	**compreesed data**

4 Experiment & Evaluation

The Star-Cubing Algorithm was implemented based on data sets from UCI website.[4]

4.1 Textbook Example Data set

To test our implementation is working correct, we first use the example data set on the textbook. It only has have four attributes(A,B,C,D). All the data-sets will run on Thinkpad X230 with 16 GB RAM and i5 processor.

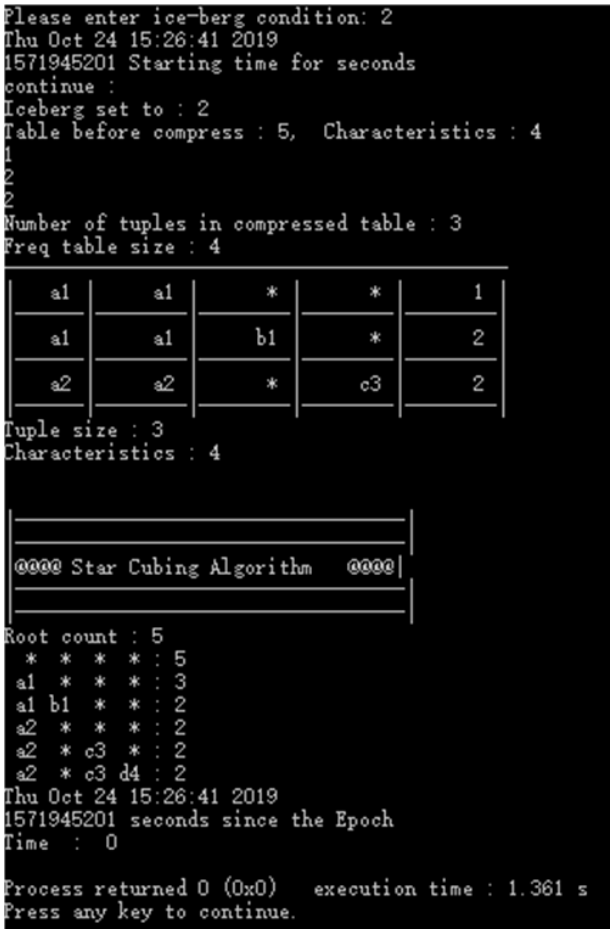


Figure 3: The result graph after Star-Cubing Algorithm.

Based on the results, our implementation works correct.

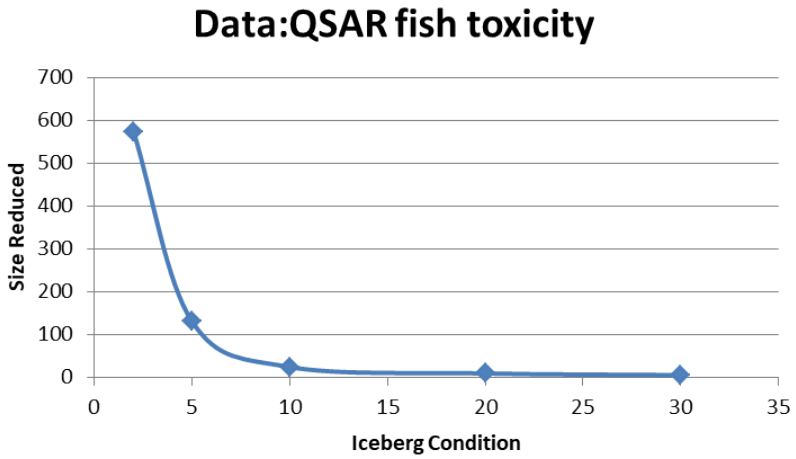
4.2 QSAR fish toxicity Data Set

This data-set containing values for 6 attributes of 908 chemicals used to predict quantitative acute aquatic toxicity towards the fish Pimephales promelas.

Data Set Characteristics:	Multivariate	Number of Instances:	908	Area:	Physical
Attribute Characteristics:	Real	Number of Attributes:	7	Date Donated	2019/9/23
Associated Tasks:	Regression	Missing Values?	N/A	Number of Web Hits:	2945

Iceberg condition	Actural size	Reduced size	Execution time
2	909	575	0.292796s
5	909	131	0.291728s
10	909	25	0.283419s
20	909	10	0.271507s
30	909	6	0.268337s

Figure 4: QSAR fish toxicity Data Set Execution parameters .



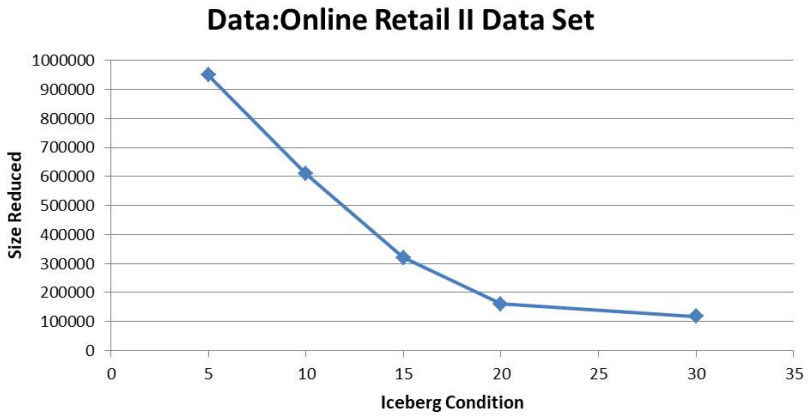
4.3 Online Retail II Data Set

This Online Retail II data set contains all the transactions occurring for a UK-based and registered, non-store online retail between 01/12/2009 and 09/12/2011.The company mainly sells unique all-occasion gift-ware. Many customers of the company are wholesalers.

Data Set Characteristics:	Multivariate, Sequential, Time-Series, Text	Number of Instances:	1067371	Area:	Business
Attribute Characteristics:	Integer, Real	Number of Attributes:	8	Date Donated	2019/9/21
Associated Tasks:	Classification, Regression, Clustering	Missing Values?	Yes	Number of Web Hits:	12184

Iceberg condition	Actural size	Reduced size	Execution time
5	1067370	949959	0.362715s
10	1067370	608400	0.301746s
15	1067370	320211	0.289419s
20	1067370	160105	0.256106s
30	1067370	117410	0.219517s

Figure 5: Online Retail II Data Set Execution parameters .



4.4 Yacht Hydrodynamics Data Set

This data set is used to predict the hydrodynamic performance of sailing yachts from their dimensions and velocities.

Data Set Characteristics:	Multivariate	Number of Instances:	308	Area:	Physical
Attribute Characteristics:	Real	Number of Attributes:	7	Date Donated	2013-01-03
Associated Tasks:	Regression	Missing Values?	N/A	Number of Web Hits:	74180

Data Set	Ice-berg	Size Before	Size After	Memery Ulsage	CPU Time
Yacht Hydrodynamics	5	538142	13616	296696	12.203s
	10	538142	6381	296696	12.109s
	20	538142	3533	296690	11.906s
	30	538142	2659	296688	11.875s

Figure 6: Yacht Hydrodynamics Data Set Execution parameters.

4.5 Wholesale customers Data Set

The data set refers to clients of a wholesale distributor. It includes the annual spending in monetary units (m.u.) on diverse product categories.

Data Set Characteristics:	Multivariate	Number of Instances:	440	Area:	Business
Attribute Characteristics:	Integer	Number of Attributes:	8	Date Donated	2014/3/31
Associated Tasks:	Classification, Clustering	Missing Values?	N/A	Number of Web Hits:	300449

Data Set	Ice-berg	Size Before	Size After	Memery UIsage	CPU Time
Wholesale customers	2	441	21	7092	0.015s
	5	441	7	7064	0.003s
	10	441	7	7056	0.003s
	30	441	7	7072	0.003s

Figure 7: Wholesale customers Data Set.

4.6 User Knowledge Modeling Data Set

It is the real data set about the students' knowledge status about the subject of Electrical DC Machines. The data set had been obtained from Ph.D. Thesis.

Data Set Characteristics:	Multivariate	Number of Instances:	403	Area:	Computer
Attribute Characteristics:	Integer	Number of Attributes:	5	Date Donated	2013/6/26
Associated Tasks:	Classification, Clustering	Missing Values?	N/A	Number of Web Hits:	111856

Data Set	Ice-berg	Size Before	Size After	Memery UI sage	CPU Time
User Knowledge Modeling Data Set	5	259	184	7000	0.000s
	10	259	17	6944	0.000s
	20	259	1	6944	0.000s
	30	259	1	6936	0.000s

Figure 8: User Knowledge Modeling Data Set.

4.7 Data for Benchmark Implementation with Number of Attributes ≤ 7

Data Set	Iceberg Condition	Size Before	Size After	Memory Usage	CPU Time (s)
Iris	5	150	140	6972	0
	10	150	48	6908	0
	20	150	4	6900	0
	30	150	1	6884	0
Car Evaluation	5	1728	1728	8504	0.031
	10	1728	1707	8488	0.031
	20	1728	1565	8388	0.016
	30	1728	1450	8328	0.015
Abalone	5	4177	3989	12016	0.078
	10	4177	3676	11772	0.078
	20	4177	3463	11668	0.078
	30	4177	3215	11500	0.078
3D Spatial Network	5	434874	25846	248600	9.671
	10	434874	13038	247980	9.641
	20	434874	4914	247996	9.531
	30	434874	2374	247976	9.531
Carbon Nanotubes	5	10722	188	15472	0.234
	10	10722	43	15468	0.234
	20	10722	43	15476	0.234
	30	10722	43	15476	0.234
BuddyMove	5	250	234	7140	0
	10	250	105	7044	0
	20	250	6	7012	0
	30	250	3	7020	0.015
Localization Data for Person Activity	5	164860	101	160712	5.484
	10	164860	101	160708	5.454
	20	164860	101	160708	5.454
	30	164860	101	160712	6.562
Parking Birmingham	5	35718	35363	35284	0.390
	10	35718	35353	35228	0.375
	20	35718	35331	35188	0.390
	30	35718	6725	21696	0.327
Istanbul Stock Exchange	5	536	7	7416	0.015
	10	536	7	7396	0.016
	20	536	1	7404	0.016
	30	536	1	7404	0.016
Banknote Authentication	5	1371	5	7744	0.031
	10	1371	1	7724	0.016
	20	1371	1	7736	0.015
	30	1371	1	7732	0.031

Algorithm was implemented for these benchmarks using Lenovo Ideapad with 8 GB RAM and i7-7500U processor. Size refers to number of instances, or tuples, within each data set and CPU time refers to the amount of time used by the processor to run the Star-Cubing Algorithm after an iceberg condition had been set.

5 Discussion

5.1 Implementation Analysis

Using the textbook data set was important to test whether or not the coded star cube algorithm worked. By replicating the same results, it is confirmed that the Star-Cubing Algorithm works for at least the textbook example. In both implementation examples, the size of the data reduces as the iceberg condition increases. This is because as the iceberg condition increases, in this case count(), the amount of tuples that meet the criteria drastically decreases. As a result, because of pruning, the execution time decreases as well. Therefore the Star-Cubing implementation for a given data set can be optimized depending upon the user's required conditions. The online retail II data set is a larger data set, having over one million instances. The execution time decrease is larger than with the smaller QSAR fish toxicity data set, which contains only about 900 instances. Where the QSAR fish toxicity data set decreases in execution time from minimum support 2 to 30 by 8.35%, the online retail II data set decreases in execution time by 39.48%. The reduced data size from the same range, however, decreases to a smaller percentage of the original size for the smaller data set compared to the larger. For the QSAR fish toxicity data the reduced size at a minimum support threshold of 30 is 0.66% of the original, whereas for the online retail II data the reduced size becomes 10.00% of the original. These results show how the effects of the Star-Cubing Algorithm is data dependent, where the amount of instances and attributes as well as the iceberg conditions have case by case different effects on the output.

The table at 4.7 provides further evidence for the data set dependent nature of the effectiveness of the Star-Cubing Algorithm. While four set iceberg conditions were used to compare the data sets, their effect varied per each benchmark. For example, for the Banknote Authentication much of the data was pruned at an iceberg value of 5. However, much less of the data is pruned when the iceberg condition is set to 1. The Parking Birmingham data has the opposite nature, where the data does not to have significant pruning until an iceberg condition of greater than 20. The figure below shows the percent decreases of the data sets used in section 4.1 with respect to each of the test iceberg conditions.

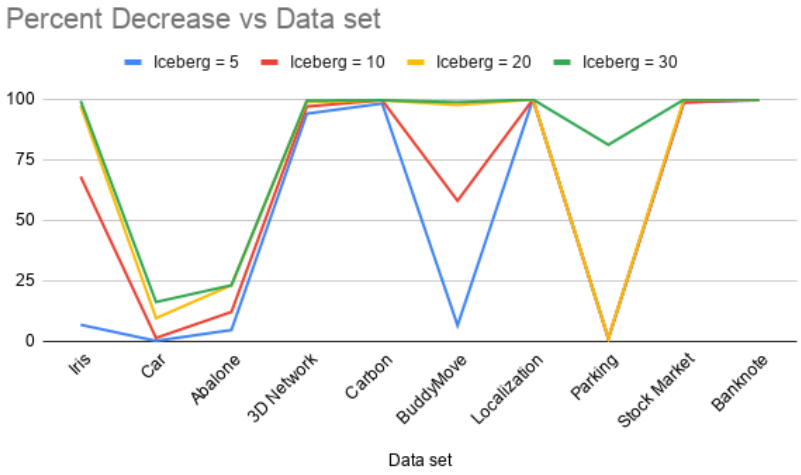


Figure 9: Percent decrease for each data set at each iceberg condition.

The general trend evident from this figure are that the percent decrease in the size of the data

increases as the iceberg condition rises, which is because as the threshold is increased the more selective the Star-Cubing Algorithm becomes. It increases the likelihood of data being pruned. There is no clear trend that transcends data sets, which demonstrates the customization element of the Star-Cubing Algorithm. The iceberg condition could be a different metric of varying values across data sets with no apparent relationship between them.

There were other trends observed as seen in the table from section 4.7. As the iceberg condition increased and the Star-Cubing Algorithm became more selective, the memory usage decreased and CPU time generally decreased. There were several exceptions, however, where the memory increased or CPU time increased after the iceberg condition increased. These cases generally occurred after the data had been pruned to a small number, which could mean that the algorithm prunes the data early on but continues the traversal process in the star tree an unnecessary amount of times. In addition, smaller data sets ($\leq$ 2000 tuples) tended to prune to a smaller percentage quicker than larger data sets. This intuitively is consistent with the process of Star-Cubing, since it is based on aggregation and pruning smaller data sets will compress quicker. However, data reduction could be achieved in preprocessing, so Star-Cubing may be more useful for large data sets.

5.2 Data Structures

Maps are used for their key,value pairing. Each key is a unique identifier which can be inserted elsewhere or deleted, but cannot be changed. However, the value connected to each key can be changed. This allows for easier access to data that can be dynamically changed.

Vectors are used for their dynamically driven nature. Their size can be changed and does not have to be initialized, which makes it effective to store data of unknown sizes. They have contiguous memory allocation, so when one memory block is unable to store elements of a vector, a new block of memory is allocated and the contents of the old memory block is copied into the new.

5.3 Star-Cubing Advantages

Star-Cubing is effective because it uses three optimization criteria. It computes multiple data cubes simultaneously, uses the smallest child for computing a more abstract data cube, and utilizes the iceberg condition for pruning. As a result, the amount of interactions between the external memory and processor is reduced as well as the memory allocated and used during the Star-Cubing process. Star-Cubing is faster and more effective than Multiway and BUC, and is an improved version of the H-tree cubing method. Just as H-tree cubing can be used to compute full data cubes, Star-Cubing can be used to compute full data cubes although the code implemented only computes the iceberg data cubes.

5.4 Star-Cubing Disadvantages

While Star-Cubing is considered more efficient than only the Multiway or BUC method, it has its limitations as well. Star-Cubing is sensitive to dimension ordering. It is recommended to organize the dimensions in decreasing cardinality to promote early pruning. This has to do with partitioning, as dimensions with many instances can be separated into more partitions, which can then be pruned. Another weakness of Star-Cubing is the size of cuboids, which are on the order of 2^n where n is the amount of dimensions. In addition, the Star-Cubing Algorithm

requires recomputation if there is a database change. Although the data structures used are dynamically driven, the actual algorithm is dependent upon the initial construction of the star table and star tree, therefore one change in the data set would require recomputation of the star table and star tree even before beginning the starcube recursion.

6 Conclusion

The Star-Cubing Algorithm provides an efficient way to compute iceberg cubes. Combining the process of Multiway aggregation and BUC pruning, Star-Cubing makes use of three optimization criteria. Its effectiveness is analyzed with several benchmark examples. The implementation of the Star-Cubing Algorithm was used to show the effect iceberg conditioning has on the data size reduction, iceberg cube computation, and memory usage/speed. The Star-Cubing Algorithm is effective in all these areas, however it is difficult to compare with other data sets due to the variability of icebergs and the data in general. A more effective way to test the productivity of the Star-Cubing Algorithm could be by comparing it to other data cube computation algorithms, such as Multiway Aggregation, BUC, and H-Tree Algorithms.

Data cube computation is an important precursor to data analysis and data mining. By computing the iceberg cuboids, the important data can be used to mine frequent patterns and perform statistical analysis. However, the Star-Cubing Algorithm can be expanded upon and further studied to provide an even deeper method of computing data cubes, such as computing cube-gradients and quotient cubes.

References

[1] Xin, D., Han, J., Li, X., & Wah, B. W. (2003, September). Star-Cubing: Computing iceberg cubes by top-down and bottom-up integration. In Proceedings of the 29th international conference on Very large data bases-Volume 29 (pp. 476-487). VLDB Endowment.

[2] Feng, Y., Agrawal, D., El Abbadi, A., & Metwally, A. (2004, April). Range Cube: Efficient cube computation by exploiting data correlation. In Proceedings. 20th International Conference on Data Engineering (pp. 658-669). IEEE.

[3] Han, J., & Kamber, M. (2012). Data Mining: C d h Concepts and Techniques.

[4] https://archive.ics.uci.edu/ml/index.php