SvedbergOpen
DISSEMINATION OF KNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: https://www.svedbergopen.com/

Research Paper Open Access

A Hybrid SLIQ–SPRINT Framework For Intelligent Faculty Workload Assignment

Benchie L. Maribao

Faculty Isabela State University, Echague, Isabela, Philippines benchie.l.maribao@isu.edu.ph

Abstract

The objective of the study focuses on the design and development of an innovation in the faculty workload system of Isabela State University. At present, the Scalable Parallelizable Induction of Decision Tree or SPRINT algorithm is being used; however, the algorithm had a weakness for classifying attributive lists and has high computational cost in calculating attribute segmentation. This study was specifically undertaken to address this flaw in the SPRINT algorithm. To do this, the SPRINT algorithm classification approach uses the SLIQ pre-sorting technique to answer the rewrites and resorts of the attribute lists.

Employing the design-based research model, the problem with the current system was grounded accordingly through personal assessment and literature study. From there, system development was done, integrating the SLIQ pre-sorting technique into the SPRINT algorithm. The enhanced system was named E-SPRINT. Initial testing indicated an improvement, specifically in the time spent in classifying attribute lists.

Keywords: Data analysis algorithm, E-SPRINT, Faculty workload system, SPRINT, SLIQ

1. Introduction

Managing the faculty workload in a sizeable university can be a complex task but the utilization of digital technology can make the process more manageable for the administrative staff. As stipulated by Liu et al. (2022), a specifically designed faculty workload system not only improves work efficiency but also ensure that human resource wastage is reduced or minimized. Moreover, Altinay (2016) also pointed out that capitalizing on digital technology transforms school management for the better which in turn will translate to the quality of services that educational institution can offer. This is in consideration to the fact that faculty manual workload management has high error rate which can affect the quality and equity of load evaluation and distribution. This issue holds much significance especially in higher education institutions where faculty members perform a variety of tasks aside from instruction such as research and even community service (AlSaeed, 2020).

Algorithms for data analysis have proven particularly useful in aiding workload management in schools. Classification analysis, for example, is widely utilized for data analysis involving the identification and assignment of categories to a data collection. As a decision support system, it employs the decision tree methodology which aids in the development of prediction algorithms for a specific variable (Song & Liu, 2015).

According to Hajjej et al. (2022), decision tree algorithms are recognized for their capability to visually aid decision-making. In addition, they are also easy to implement and interpret; are applicable to different types of variables; and produces reliable results. Similarly, Zhang et al. (2020) concur that the decision tree algorithm is one of the most commonly used data-mining methods, Because of this, it has underwent numerous transformations throughout time so as to match the needs of its users. Ultimately, it is characterized by its top-down recursive division and its essentially greedy basic algorithm. Its flowchart structure has internal nodes that test a specific attribute; branches that represent possible outcomes of these tests; and leaf nodes denoting the classification result (Dai et al., 2016). Building a decision tree algorithm entails two stages: tree growth and tree pruning. The top-down approach to tree growth involves repeatedly partitioning the tree until all the data items are labeled with the same class. This is computationally tasking and intensive since the training dataset is

traversed repeatedly until the correct decision set is determined. Meanwhile, in the pruning phase, a bottom-up approach is used to improve the prediction and classification accuracy of the algorithm; thus minimizing the possibility of overfitting as this results to misclassification error (Mehta et al., 1996; Kotsiantis, 2007; Steinberg, 2009).

The Scalable Parallelizable Induction of Decision Tree or SPRINT algorithm was created as a fast scalable decision tree classifier. Implemented in serial and parallel patterns for data placement and load balancing, it was designed to handle continuous and categorical attributes, enabling multiple processors to work on building a single model (Anyawu & Shiva, 2009). This makes SPRINT ideal for data-mining. Despite its usability, the algorithm had a weakness for classifying attributive lists and its high computational cost in calculating attribute segmentation (Wang et al., 2016).

This study was specifically undertaken to address this flaw in the SPRINT algorithm. The SPRINT algorithm's classification phase involves rewriting and resorting attribute lists; to address these issues, the SLIQ pre-sorting technique is employed. Thus, the purpose of this research is to test how well the improved SPRINT algorithm performs in terms of time when applying it to attribute list classifications.

METHODOLOGY

Research Design

The enhanced decision tree classification algorithm of the faculty workload management system of Isabela State University was developed and evaluated using the Design-Based Research (DBR) method. The choice of Design-Based Research is based on the fact that it offers a systematic method to the generation, development, testing and improvement of technology breakthroughs to solve real-world institutional challenges. The researcher might analyze and evaluate the existing SPRINT algorithm in iterations. Identify the limits of the existing SPRINT algorithm and develop a better one.

The study focused on improving the Scalable Parallelizable Induction of Decision Tree (SPRINT) algorithm by incorporating the Supervised Learning in Quest (SLIQ) pre-sorting method to minimize computing expenses and enhance the efficiency of attribute-list classification.

Research Environment

The study was undertaken within the context of the faculty workload management system of Isabela State University. The system holds faculty workload data including teaching assignments, faculty rank, designation, number of subject units, Full-Time Equivalent (FTE) and gender. The performance of the developed algorithm was simulated and evaluated using synthetic and sampled faculty workload datasets.

System Development Procedure

The development of the enhanced algorithm was conducted in three major phases:

1. Grounding of Data

This phase involved identifying the existing problems and limitations of the conventional SPRINT algorithm through:

- Literature review;
- Personal assessment of the current faculty workload system; and
- Analysis of existing decision-tree classification techniques.

The review revealed that the SPRINT algorithm experiences:

- High computational cost during attribute segmentation;
- Repeated rewriting and resorting of attribute lists during tree growth; and
- Increased processing time when handling large datasets.

These identified issues became the basis for integrating the SLIQ pre-sorting technique into the SPRINT algorithm.

2. System Development

The enhanced algorithm, referred to as E-SPRINT, was developed by integrating the SLIQ pre-sorting mechanism into the growth phase of the SPRINT algorithm before the evaluation of splitting points.
The enhancement process involved the following functions:

a. Pre Sorting Function

The training dataset was pre-sorted once before tree construction. Separate attribute lists and class lists were created to eliminate repeated sorting operations during node splitting.
The training dataset consisted of the following attributes:

- Faculty name;
- Rank;
- Designation;
- Number of subject units;
- Full-Time Equivalent (FTE); and
- Gender.

The pre-sorting mechanism organized the attribute values and linked them to corresponding class labels to facilitate faster split evaluation.

b. Evaluate Split Function

The EvaluateSplit function determined the optimal splitting point for each node using the **Gini Index**. The algorithm compared attribute-list indexes with class-list indexes to identify the best split point with the smallest Gini Index value.
The process recursively evaluated:

- Root-node splitting;
- Left-leaf node splitting; and
- Right-leaf node splitting.

c. Make Tree Function

The MakeTree function constructed the decision tree recursively based on the identified splitting points. The algorithm partitioned the dataset into leaf nodes until the optimal classification structure was achieved.

Data Collection

The study utilized synthetic datasets derived from faculty workload records. Ten sample records were initially used to simulate the classification process and validate the functionality of the enhanced algorithm.
For performance evaluation, datasets containing:

- 500 records,
- 700 records, and
- 1,000 records

were built and used to compare the efficiency of the conventional SPRINT method with that of the proposed E-SPRINT technique.

Performance Evaluation

The effectiveness of the improved system has been evaluated using the following criteria:
1. Time of classification: The time required to categorize attribute lists was measured and compared between the SPRINT and E-SPRINT approaches.
2. Computational Cost: The study analyzed the computing cost of sorting and partitioning.
3. Decision tree efficiency: The efficiency of the generated decision tree was tested in terms of database growth and computing complexity.

RESULTS AND DISCUSSION

Grounding of Data

Personal assessment as well as a comprehensive literature review revealed the issues that needed to be addressed. Classification, albeit a commonly applied data mining technique, is complex. The SPRINT method can handle both continuous and categorical features, and can be parallelized in the sense that numerous processors can work together to form a single model.
The scalability of this parallelization is demonstrated to its users through its speeding up and scaling up properties. While SPRINT's combination of attributes makes it a suitable data mining tool, its inadequacy in categorizing attribute lists causes it to repeatedly rewrite and resort. This increases the size of the database and also increases the computational cost while sorting. Rewriting the attribute list The SPRINT takes time to clasify attribute lists.

Given this, the issues that need to be addressed are within the tree growth phase: (1) time and (2) analyzing large data base to address parallelization. Other studies have also detailed these issues as follows:

Table 1. Summary of Problems Identified in SPRINT Classification

Study	Problem Identified
A Searching Method Candidates Segmentation Point in SPRINT Classification (Wang, et al., 2016)	It is high in Computational Cost in the calculation of attribute segmentation (Gini Index).
Improved SPRINT Algorithm and Its Application in the Physical Data Analysis (Ding & Zheng, 2014)	The classification process for the attribute lists is time consuming and space complexity. (Selection of splitting point)
The Research of Decision Tree Mining Based on Hadoop (Sivasandeep & Reddy, 2014)	The decision tree mining calculation is extremely huge for the single node massive data (Parallelization)
Research on SPRINT Algorithm in Cloud Computing (Zhang, 2010)	Time consuming on analyzing mass data (Parallelization)
Improved SPRINT Algorithm and Its Research under Distributed Environment (Yu et al., 2008)	It utilizes numerous attribute lists for data storage, consuming system resources; splits are executed through the establishment of a hash table, and the node splitting procedure is somewhat intricate.
Study on the parallelism of decision tree classification based on SPRINT (Wei, 2005)	Decision tree construction on huge data sets is computationally prohibitive due to the complexities, which are often proportional to the size of the training data set.
Clouds (Haider & Asghar, 2013; Alsabti et al., 1998)	The rewrites and resorts of Attribute lists during classification process
BOAT (Haider & Asghar, 2013; Gehrke et al., 1999)	The rewrites and resorts of Attribute lists during classification process
Rainforest (Haider & Asghar, 2013; Gehrke et al., 1998)	The rewrites and resorts of Attribute lists during classification process
Public (Haider & Asghar, 2013; Rajeev & Kyuseok, 1998)	The rewrites and resorts of Attribute lists during classification process (Gini index)

A scalable decision tree approach known as Supervised Learning in Quest (SLIQ) can be applied in either a serial or parallel fashion, and it is quick. During the tree-building phase, it incorporates a pre-sorting technique and a breadth-first greedy strategy to recursively partition training data sets. As a first step, SLIQ employs a scheme to do away with data sorting at each decision tree node. Both numerical and categorical attributes can be handled by SLIQ when constructing a decision tree model (Phyu, 2009).

As per literature analysis, it was determined that SLIQ has not been used to solve the primary problems of SPRINT which is the length of its classification process. The following describes how this can be addressed using SLIQ:

1. Pre-Sorting Technique

The sorting time is the principal criterion for determining the optimal split in a decision tree node. The primary method employed in SLIQ is the removal of data resorts at each node. Instead, the training data are merely sorted once at the commencement of the tree development phase. To achieve this pre-sorting methodology, the subsequent data structures are employed: A distinct list (class list) for each attribute is generated in the training data, serving as the class label associated with the examples. An entry in an attribute list comprises two components: an attribute value and an index to the class list. Likewise, the class lists entry comprises two components: a class name and a reference to a leaf node of the decision tree. The i^{th} entry of the class list in the training data corresponds to the i^{th} example. Every leaf node signifies a division of the training data. The partition is determined by the amalgamation of the predicates along the path from the node to the root. Consequently, the class list can ascertain the partition to which an example is assigned. The graphic below illustrates the data structures prior to and subsequent to pre-sorting (Mehta et al., 1996).

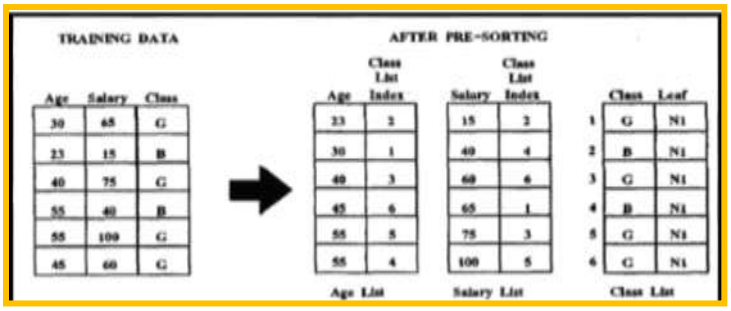


Figure 1. SLIQ Pre-sorting Output

In order to handle the rewrites and resorts of the attribute lists in the SPRINT algorithm's classification phase, the SLIQ pre-sorting strategy is suggested. This is justified by the fact that SLIQ recursively splits the training dataset using a breadth-first greedy method that incorporates pre-sorting techniques during the growth phase. The strategy employs a method that eradicates data-sorting at every node of the decision tree. This decreases expenses associated with the assessment of quantitative qualities. Consequently, a decision tree model capable of managing both numeric and categorical attributes is produced.

2. Decision Support System (DSS)

A Decision Support System (DSS) is a computer-based information system that facilitates decision-making processes within businesses or organizations, sometimes leading to the ranking, sorting, or selection of alternatives. DSS serve the management, operations, and planning levels of an organization (usually mid and higher management) and help people make decisions about problems arising (Zhang & Babovic, 2011).

While academics have perceived DSS as a tool to support decision-making process, DSS users see DSS as a tool to facilitate organizational processes (Keen, 1980).

System Development

The development of the improved algorithm was based on the pre-sorting technique of SLIQ to offset the identified limitation of SPRINT. The developed algorithm was integrated into the growth phase of SPRINT before the evaluation of splitting point on each node. It was named E-SPRINT Algorithm. The process for the enhancement of SPRINT algorithm is shown in Figure 2.

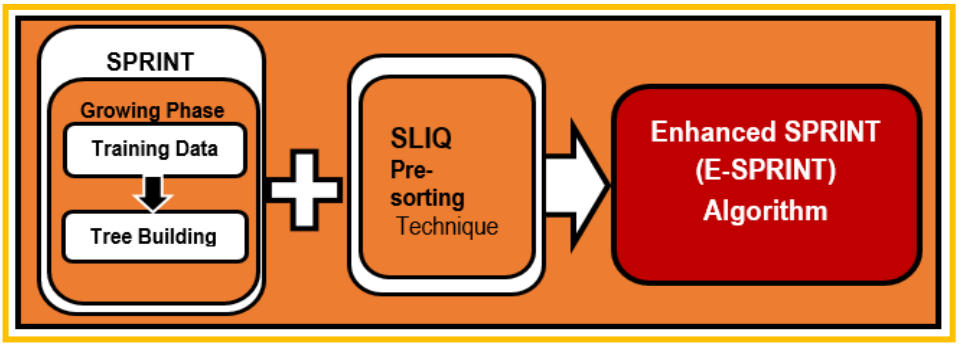


Figure 2. Enhancement of SPRINT Algorithm with the integration of SLIQ Pre-sorting Technique

Simulation of the E-SPRINT Algorithm

The E-SPRINT algorithm encompasses three functions: Pre-Sorting, EvaluateSplit, and MakeTree. Ten records, retrieved and randomly selected from the database of faculty workloads at Isabela State University, were utilized in the simulations to evaluate the categorization process of the E-SPRINT method.

1. Pre-Sorting Function

Training Data: The training data comprised of six attributes: name, rank, designation, number of units, FTE, and gender. The below figure shows the pre-sorting output:

Figure 3. Pre-sorting Technique Output

2. Evaluate Split

The Evaluate Split () function assesses the optimal split point. The result of the pre-sorting method will serve as the foundation. The root node is designated as N1. To ascertain the optimal split point for the initial division, the Rank List should be considered first. It served as the foundation for the initial division of the qualities. The class index of the Rank List was compared and aligned with the Class List index to ascertain the position of the attribute value. The Gini Index was employed to assess the optimal split point for an attribute. Upon obtaining the Gini index for each record, identify the minimum Gini index. The minimum value is 0.417, found at Position 6, and the calculated splitting point is 17. The left-side split leaf node is designated as N2, whereas the opposing node is referred to as N3. The computational expense of the initial division is as follows:

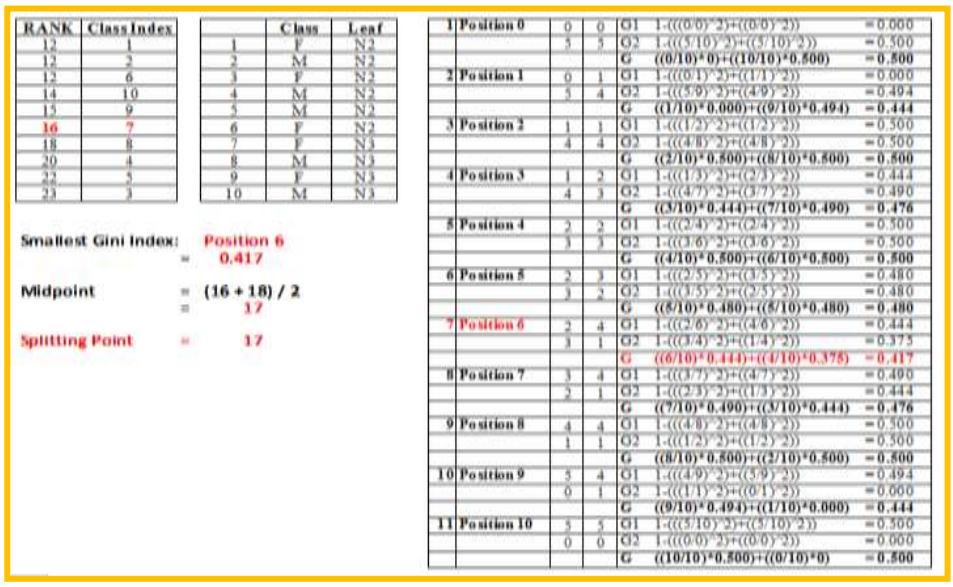
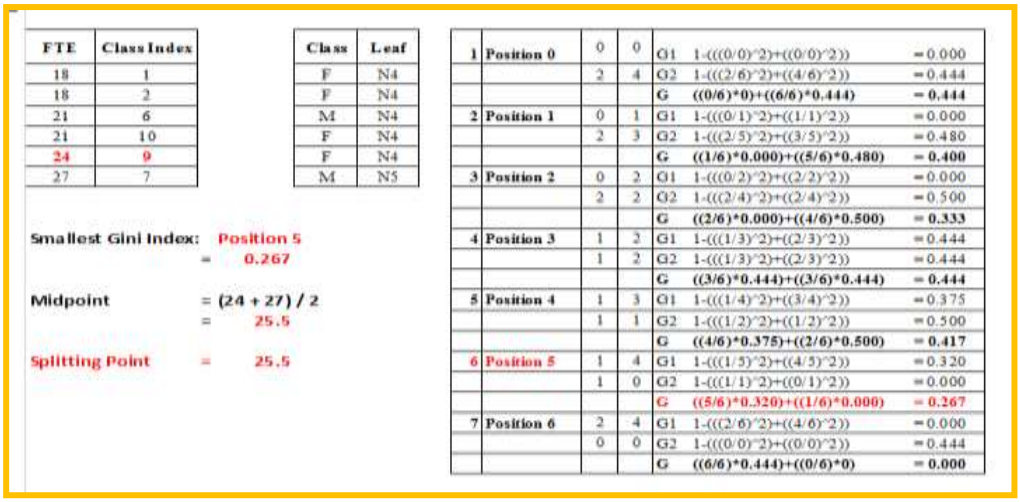


Figure 4. First Split Computational Cost

The second split is contingent upon the results of the initial division. To ascertain the optimal split point, the identical procedure employed for the initial split was replicated; however, this time, the FTE List was taken into account. N2 (left-leaf node) served as the foundation for the computational expense associated with identifying the optimal split point for the subsequent division, designated as N4 (left-leaf node) and N5 (right-leaf node). The calculated minimum Gini index is 0.267, identified at Position 5 with a splitting point value of 25.5. N3 (right-leaf node) served as the foundation for the computational expense associated with identifying the optimal split point for the subsequent division, designated as N6 (left-leaf node) and N7 (right-leaf node). The calculated minimum Gini index is 0.000, found at Position 3 with a splitting point value of 35. The computational expense of the second division is illustrated hereunder through Figures 5 and 6, respectively:



FTE	Class Index
18	1
18	2
21	6
21	10
24	9
27	7

Class	Leaf
F	N4
F	N4
M	N4
F	N4
F	N4
M	N5

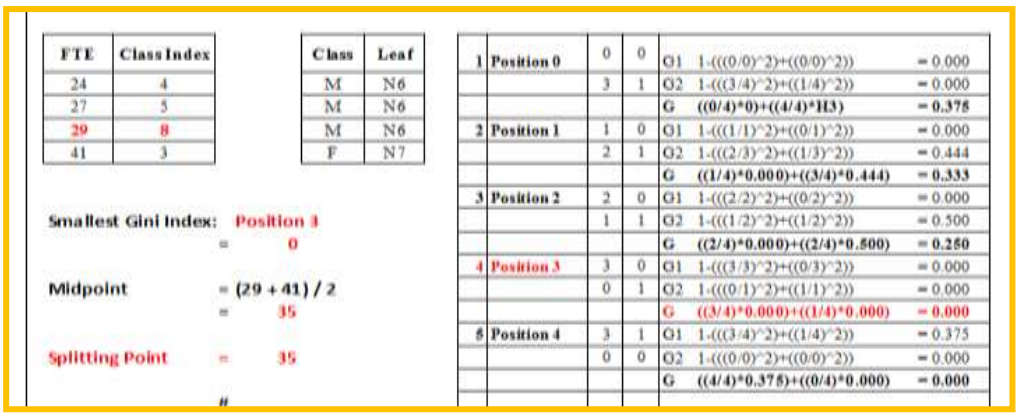
Smallest Gini Index: Position 5 = 0.267

Midpoint = (24 + 27) / 2 = 25.5

Splitting Point = 25.5

1	Position 0	0	0	G1	1-(((0/0)^2)+((0/0)^2))	= 0.000
		2	4	G2	1-(((2/6)^2)+((4/6)^2))	= 0.444
				G	((0/6)*0)+((6/6)*0.444)	= 0.444
2	Position 1	0	1	G1	1-(((0/1)^2)+((1/1)^2))	= 0.000
		2	3	G2	1-(((2/5)^2)+((3/5)^2))	= 0.480
				G	((1/6)*0.000)+((5/6)*0.480)	= 0.400
3	Position 2	0	2	G1	1-(((0/2)^2)+((2/2)^2))	= 0.000
		2	2	G2	1-(((2/4)^2)+((2/4)^2))	= 0.500
				G	((2/6)*0.000)+((4/6)*0.500)	= 0.333
4	Position 3	1	2	G1	1-(((1/3)^2)+((2/3)^2))	= 0.444
		1	2	G2	1-(((1/3)^2)+((2/3)^2))	= 0.444
				G	((3/6)*0.444)+((3/6)*0.444)	= 0.444
5	Position 4	1	3	G1	1-(((1/4)^2)+((3/4)^2))	= 0.375
		1	1	G2	1-(((1/2)^2)+((1/2)^2))	= 0.500
				G	((4/6)*0.375)+((2/6)*0.500)	= 0.417
6	Position 5	1	4	G1	1-(((1/5)^2)+((4/5)^2))	= 0.320
		1	0	G2	1-(((1/1)^2)+((0/1)^2))	= 0.000
				G	((5/6)*0.320)+((1/6)*0.000)	= 0.267
7	Position 6	2	4	G1	1-(((2/6)^2)+((4/6)^2))	= 0.000
		0	0	G2	1-(((0/0)^2)+((0/0)^2))	= 0.444
				G	((6/6)*0.444)+((0/6)*0)	= 0.000

Figure 5. Second Split (Left Node) Computational Cost



FTE	Class Index
24	4
27	5
29	8
41	3

Class	Leaf
M	N6
M	N6
M	N6
F	N7

Smallest Gini Index: Position 3 = 0

Midpoint = (29 + 41) / 2 = 35

Splitting Point = 35

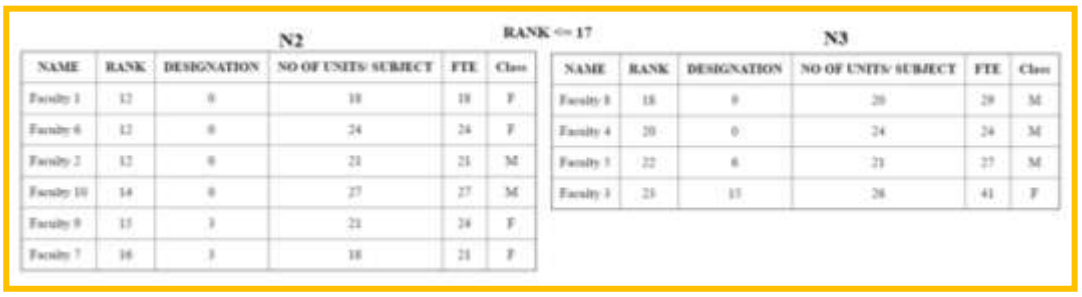
1	Position 0	0	0	G1	1-(((0/0)^2)+((0/0)^2))	= 0.000
		3	1	G2	1-(((3/4)^2)+((1/4)^2))	= 0.000
				G	((0/4)*0)+((4/4)*H3)	= 0.375
2	Position 1	1	0	G1	1-(((1/1)^2)+((0/1)^2))	= 0.000
		2	1	G2	1-(((2/3)^2)+((1/3)^2))	= 0.444
				G	((1/4)*0.000)+((3/4)*0.444)	= 0.333
3	Position 2	2	0	G1	1-(((2/2)^2)+((0/2)^2))	= 0.000
		1	1	G2	1-(((1/2)^2)+((1/2)^2))	= 0.500
				G	((2/4)*0.000)+((2/4)*0.500)	= 0.250
4	Position 3	3	0	G1	1-(((3/3)^2)+((0/3)^2))	= 0.000
		0	1	G2	1-(((0/1)^2)+((1/1)^2))	= 0.000
				G	((3/4)*0.000)+((1/4)*0.000)	= 0.000
5	Position 4	3	1	G1	1-(((3/4)^2)+((1/4)^2))	= 0.375
		0	0	G2	1-(((0/0)^2)+((0/0)^2))	= 0.000
				G	((4/4)*0.375)+((0/4)*0.000)	= 0.000

Figure 6. Second Split (Right Node) Computational Cost

3. Make Tree

The MakeTree ( ) function covers the splitting of attribute for each leaf nodes. The splitting of attribute for each leaf nodes was based on the result of the EvaluateSplit ( ) function.

The splitting point of N1 (root node) is 17. It was based on the computational cost. Based on the splitting point of N1 in which RANK < =17, there are 6 records that belong to N2 (leaf node) and 4 records to the N3 (leaf node). The first split output is as follows:



N2 RANK <= 17 N3

NAME	RANK	DESIGNATION	NO OF UNITS/ SUBJECT	FTE	Class
Faculty 1	12	0	18	18	F
Faculty 6	12	0	24	24	F
Faculty 2	12	0	21	21	M
Faculty 10	14	0	27	27	M
Faculty 9	15	1	21	24	F
Faculty 7	16	1	18	21	F

NAME	RANK	DESIGNATION	NO OF UNITS/ SUBJECT	FTE	Class
Faculty 8	18	0	20	20	M
Faculty 4	20	0	24	24	M
Faculty 5	22	0	21	27	M
Faculty 3	23	13	28	41	F

Figure 7. First Split Output

The second split relied on the results of the initial split. N2 was first split. Six records pertain to N2. The computational expense denotes splitting threshold at 25.5. Records with an FTE of 25.5 or below (FTE ≤ 25.5) will be clasified under N4 (leaf node), while the other records will be classified under N5 (leaf node). There are five records on N4 and one record on N5. The bifurcation point of N3 is 35. N3 comprised 4 records. According

to splitting criterion where FTE is less than or equal to 35, there are three records associated with N6 (leaf node) and one record associated with N7 (leaf node). The second split output is displayed below:

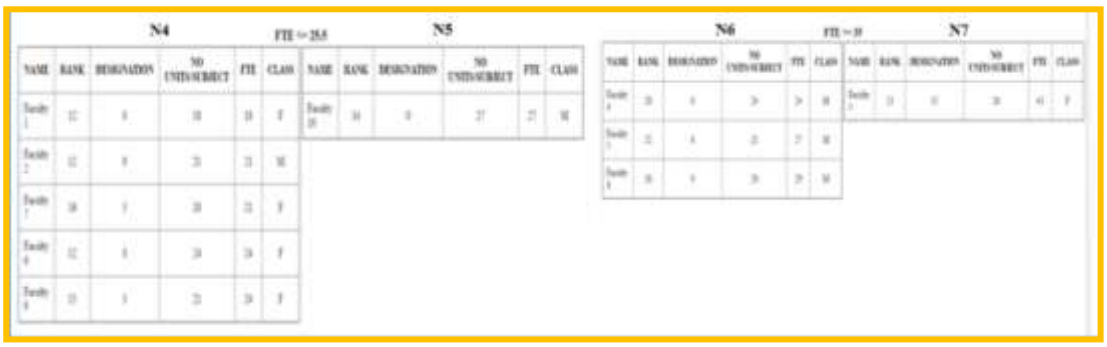


Figure 8. Second Split Output

Testing and Preliminary Evaluation

Table 2. Time for Attribute List Classification

Dataset	SPRINT Algorithm	E-SPRINT Algorithm
500 records	0.10523 s	0.06367 s
700 records	0.14670 s	0.09852 s
1,000 records	0.17953 s	0.12624 s

Using a synthetic datasets of faculty workloads classifying records in 500, 700 and 1000 records, results showed that SPRINT recorded a longer time to classify records. It was shown in 1000 records, SPRINT has to classify records in 0.17953s as compared with E-SPRINT with 0.12624s. It proves therefore that E-SPRINT Algorithm is much faster and records a shorter period of time in classification.

The developed algorithm based on SLIQ pre-sorting technique in terms of growing and splitting attributes shows that there is no growth in the size of the database as well as there is no increase in sorting computational cost. The integration of SLIQ pre-sorting techniques removed the process of rewriting and resorting to its initial training data set. In addition, the Enhanced SPRINT Algorithm rebuilds its tree from the splitting nodes. This recorded a faster time in classifying attribute lists.

REFERENCES

1. Alsabti, K., Ranka, S. & Singh, V. (1998). CLOUDS: A Decision Tree Classifier for Large Datasets. Proceedings of 4th International Conference on Knowledge Discovery and Data Mining
2. Al Saeed, D. (2020). Toward achieving quality in faculty-load allocation: A developed faculty-load-management system. International Journal for Quality Research, 14(4), 1191-1206. https://doi.org/10.24874/ijqr14.04-13
3. Altinay, F., Dagli, G., & Altinay, Z. (2016). Digital transformation in school management and culture. Virtual Learning. https://doi.org/10.5772/65221
4. Anyawu, M. N., & Shiva, S. G. (2009). Comparative analysis of serial decision tree classification algorithms. Matthew N. Anyanwu & Sajjan G. Shiva International Journal of Computer Science and Security, 3(3), 230-240.
5. Dai, Q., Zhang, C., & Wu, H. (2016). Research of decision tree classification algorithm in data mining. International Journal of Database Theory and Application, 9(5), 1-8. https://doi.org/10.14257/ijdta.2016.9.5.01
6. Ding, Y., & Zeng, Z. (2014). Improved SPRINT Algorithm and Its Application in the Physical Data Analysis. China Sport Science;2014-06. The Lab of Sports Science of Human Body,Xinjiang Normal University;College of Physical Education,Xinjiang Normal University;School of Software and Microelectronics,Peking University
7. Gehrke, J., Ganti, V. & Ramakrishnan, R. (1999). BOAT: A Framework for Fast Decision Tree Construction of Large Datasets. Proceedings of the 24th VLDB Conference. New York. USA

8. Haider, A.A., and Asghar, S. (2013). A Survey of Logic Based Classifiers. International Journal of Future Computer and Communication, Vol. 2, No.2, April 2013

9. Hoadley, C., & Campos, F. C. (2022). Design-based research: What it is and why it matters to studying online learning. Educational Psychologist, 57(3), 207-220. https://doi.org/10.1080/00461520.2022.2079128
10. Keen, P.(1980). Decision support systems : a research perspective. Cambridge, Mass. : Center for Information Systems Research, Alfred P. Sloan School of Management.http://hdl.handle.net/1721.1/47172
11. Kotsiantis, S. (2007). Supervised Machine Learning: A Review of Classification Techniques. Informatica, 31. 249-268.
12. Liu, Q., Zhang, H., Fan, L., Li, T., & Jin, X. (2022). Design and implement of University teachers' workload management system. Journal of Physics: Conference Series, 2278(1), 012012. https://doi.org/10.1088/1742-6596/2278/1/012012
13. Mehta, M., Agrawal, R., & Rissanen, J. (1996, January). SLIQ: A fast scalable classifier for data mining [Paper presentation]. Advances in Database Technology - EDBT'96, 5th International Conference on Extending Database, Avignon, France.
14. Phyu, T. N. (2009). Survey of Classification Techniques in Data Mining. Proceedings of the International Multiconference of Engineers and Computer Scientists 2009 Vol I, IMECS 2009, March 8-20, 2009 Hong Kong.
15. Rajeev, R. & Kyuseok, S. (1998). PUBLIC: A Decision Tree Classifiers that Integrates Building and Pruning. Proceedings of the 24th VLDB Conference, New York, USA.
16. Shah, J.K., Ensminger, D.C., & Thier, K.S. (2015). The Time for Design-Based Research Is "Right" and "Right Now". Mid-Western educational researcher, 27, 152-171.
17. Sivasandeep, B. & Reddy, B. (2014). The Research on Decision Tree Mining Based on Hadoop. International Journal of Computer and Organization Trends. Vol. 8. Number 2, May 2014. http:/www.ijcotjournal.org
18. Song, Y., & Liu, Y. (2015). Decision tree methods: applications for classification and prediction. Shanghai Archives for Psychiatry, 27(2), 130–135. https://doi.org/10.11919/j.issn.1002-0829.215044
19. Steinberg, D. (2009). The Top Ten Algorithms in Data Mining," Ch 10, Taylor and Francis Group, LLC, 2009.
20. Wang, Z., Wang, J., Huo, Y., Tuo, Y., & Yang, Y. (2016). A Research Method of Candidate Segmentation Point in SPRINT Classification. Journal of Electrical and Computer Engineering Vol 2016. Article ID 2168478. Hindawi Publishing Corporation. http://dx.doi.org.
21. Wang, Z., Wang, J., Huo, Y., Tuo, Y., & Yang, Y. (2016). A searching method of candidate segmentation point in SPRINT classification. Journal of Electrical and Computer Engineering, 2016, 1-5. https://doi.org/10.1155/2016/2168478
22. Wei, H. (2005). Study on the parallelism of decision tree classification based on SPRINT. Journal of Computer Applications January 2005. Administrative Office, Southwest Jiaotong University, Chengdu Sichuan 610031, China.
23. Yu, L., Gao, Y., & Tian, Y. (2008). Improved SPRINT Algorithm and Its Research under Distributed Environment. Journal of Jilin University (Science Edition); 2008-06. College of Computer Science and Technology, Jilin University, Changchun 130012, China
24. Zhang, C. (2010). Research on SPRINT Algorithm in Cloud Computing. Computer Engineering and Software; 2010-11. Computer Science and Technology School, cumt, xuzhou.221116 China
25. Zhang, S. & Babovic, V. (2011). An evolutionary real options framework for the design and management of projects and systems with complex real options and exercising conditions". Decision Support Systems. 51 (1): 119–129.
26. Zhang, Z., Zhao, Z., & Yeom, D. (2020). Decision tree algorithm-based model and computer simulation for evaluating the effectiveness of physical education in universities. Complexity, 2020, 1-11. https://doi.org/10.1155/2020/8868793