ISSN 2389-8186
E-ISSN 2389-8194
Vol.7, No. 2-1
Julio-diciembre de 2020
doi: https://doi.org/10.16967/23898186.667

ARTÍCULOS ORIGINALES

Comparative Analysis of Decision Tree Algorithms for Data Warehouse Fragmentation*

pp. 31-43

NIDIA RODRÍGUEZ MAZAHUA**
LISBETH RODRÍGUEZ MAZAHUA***
ASDRÚBAL LÓPEZ CHAU****
GINER ALOR HERNÁNDEZ*****

* The authors are very grateful to Tecnológico Nacional de México for supporting this work. Also, this research paper was sponsored by the CONACYT.

** Mtra. in Administrative Engineering. Tecnológico Nacional de México, Veracruz, México. E-mail: dci.nrodriguez@ito-depi.edu.mx. ORCID: 0000-0002-8227-9476. Google Scholar: https://scholar.google.es/citations?hl=es&user=27GDu_gAAAAJ.

*** PhD in Computer Science. Tecnológico Nacional de México, Veracruz, México. E-mail: lrodriguezm@ito-depi.edu.mx. ORCID: 0000-0002-9861-3993. Google Scholar: https://scholar.google.es/citations?user=2hZw4HAAAAAJ&hl=es.

**** PhD in Computer Science. Centro Universitario UAEM Zumpango, Zumpango, México. E-mail: alchau@uaemex.mx. ORCID: 0000-0001-5254-0939. Google Scholar: https://scholar.google.es/citations?user=FqdMAaQAAAAJ&hl=es&oi=sra.

***** PhD of Science in the specialty of Electrical Engineering. Tecnológico Nacional de México, Veracruz, México. E-mail: galorh@orizaba.tecnm.mx. ORCID: 0000-0003-3296-0981. Google Scholar: https://scholar.google.es/citations?hl=es&user=8Z-gf4KwAAAAJ.

CC BY NC SA

ISSN 2389-8186
E-ISSN 2389-8194

Vol.7, No. 2-1
Julio-diciembre de 2020
doi: https://doi.org/10.16967/23898186.667

COMO CITAR ESTE ARTÍCULO
How to cite this article:

Rodríguez, N. et al. (2020). Comparative Analysis of Decision Tree Algorithms for Data Warehouse Fragmentation. *Revista Perspectiva Empresarial*, *7*(2-1), 31-43.

Recibido: 20 de agosto de 2020
Aceptado: 07 de diciembre de 2020

ABSTRACT One of the main problems faced by Data Warehouse designers is fragmentation. Several studies have proposed data mining-based horizontal fragmentation methods. However, not exists a horizontal fragmentation technique that uses a decision tree. This paper presents the analysis of different decision tree algorithms to select the best one to implement the fragmentation method. Such analysis was performed under version 3.9.4 of Weka, considering four evaluation metrics (Precision, ROC Area, Recall and F-measure) for different selected data sets using the Star Schema Benchmark. The results showed that the two best algorithms were J48 and Random Forest in most cases; nevertheless, J48 was selected because it is more efficient in building the model.

KEY WORDS Data analysis, computer systems, databases, artificial intelligence, decision making.

Análisis comparativo de algoritmos de árboles de decisión para la fragmentación de almacenes de datos

RESUMEN Uno de los principales problemas a los que se enfrentan los diseñadores de almacenes de datos es la fragmentación. Varios estudios han propuesto métodos de fragmentación horizontal basados en la minería de datos. No obstante, no existe una técnica de fragmentación horizontal que utilice un árbol de decisión. Este trabajo presenta el análisis de diferentes algoritmos de árboles de decisión con el fin de seleccionar el mejor para implementar el método de fragmentación. Dicho análisis se realizó bajo la versión 3.9.4 de Weka, considerando cuatro métricas de evaluación (Precision, ROC Area, Recall y F-measure) para diferentes conjuntos de datos seleccionados utilizando el Star Schema Benchmark. Los resultados mostraron que los dos mejores algoritmos fueron J48 y Random Forest en la mayoría de los casos; sin embargo se seleccionó J48 por ser más eficiente en la construcción del modelo.

PALABRAS CLAVE análisis de datos, sistemas informáticos, bases de datos, inteligencia artificial, toma de decisiones.

Análise comparativa de algoritmos de árvores de decisão para a fragmentação de armazéns de dados

RESUMO Um dos principais problemas aos que se enfrentam os desenhadores de armazéns de dados é a fragmentação. Vários estudos hão proposto métodos de fragmentação horizontal baseados na mineração de dados. Não obstante, não existe uma técnica de fragmentação horizontal que utilize uma árvore de decisão. Este trabalho apresenta a análise de diferentes algoritmos de árvores de decisão com o fim de selecionar o melhor para implementar o método de fragmentação. Dita análise se realizou sob a versão 3.9.4 de Weka, considerando quatro métricas de avaliação (Precision, ROC Area, Recall e F-measure) para diferentes conjuntos de dados selecionados utilizando o Star Schema Benchmark. Os resultados mostraram que os dois melhores algoritmos foram J48 e Random Forest na maioria dos casos; entretanto se selecionou J48 por ser mais eficiente na construção do modelo.

PALAVRAS CHAVE análise de dados, sistemas informáticos, bases de dados, inteligência artificial, toma de decisões.

Introduction

A Data Warehouse —DW— is a theme-oriented, integrated, time variable and non-volatile data collection in support of management's decision-making process. Data warehousing provides architectures and tools for business executives to systematically organize, understand and use the data to make strategic decisions. Data warehousing systems are valuable tools in today's fast-changing and competitive world. In recent years, many companies have spent millions of dollars building company-wide data warehouses. Many people feel that with increasing competition across industries, data warehousing is the newest indispensable marketing strategy and a retention of customers way by learning more about their needs (Han, Kamber and Pei, 2012).

On the other hand, fragmentation is a distributed database design technique that consists of dividing each database relation in smaller fragments and treating each fragment as an object in the database separately, there are three alternatives for that: horizontal, vertical and hybrid fragmentation (Ozsu and Valduriez, 2020).

One of the main problems faced by DW designers is fragmentation. Several studies have proposed data mining-based horizontal fragmentation methods, which focus on optimizing query response time and execution cost to make the DW more efficient. However, to the best of our knowledge there not exists a horizontal fragmentation technique that uses a decision tree to carry out fragmentation. Since decision tree classifiers are so popular because their construction does not require any domain knowledge or parameter setting, they can handle multidimensional data, the learning and classification steps of decision tree induction are simple and fast, and they have good accuracy (Han, Kamber and Pei, 2012), and given the importance of decision trees in classification, since they allow obtaining pure partitions (subsets of tuples) in a data set using measures such as Information Gain, Gain Ratio and the Gini Index, the aim of this work is to use decision trees in the DW fragmentation. This paper presents the analysis of different decision trees algorithms to select the best one to implement the fragmentation method performed under version 3.9.4 of Weka, considering four evaluation metrics (Precision, ROC Area, Recall and F-measure) for different selected data sets, using the Star Schema Benchmark —SSB— (Star Scheme Benchmark).

This paper is made up of the following parts: (i) the introduction; (ii) a review of related works on DW horizontal fragmentation; (ii) the methodology used in this work for the analysis of decision tree algorithms and a description of each algorithm is given; (iii) reports the preliminary results in the work, and finally (iv) the article is concluded and the future work.

Related Works

Cloud SDW (Spatial DW) and spatial OLAP (On-line Analytical Processing) as a Service concepts were presented in Costa et al. (2016). Later those concepts were used to describe two different hierarchy-based data partitioning techniques for the SDW hosted in the cloud: Spatial-based partitioning and Conventional-based partitioning. In contrast, the approach proposed by Ettaoufik and Ouzzif (2017), consisted of an incremental horizontal fragmentation technique for the DW through a web service. The goal was to automate the implementation of incremental fragmentation in order to optimize a new query load.

In Barkhordari and Niamanesh (2018), it was proposed a method called Chabok, which uses two phase Map-Reduce to solve DW problems with big data. Chabok fragments horizontally the fact table. If there are homogeneous nodes, the same number of records is allocated to each Fact-Mapper node. As part of their ongoing work on workload-driven partitioning (Boissier and Kurzynski, 2018), implemented an approach called aggressive data skipping and extended it to handle both analytical and transactional access patterns. The authors evaluated their approach with the workload and data of a production system of a global 2000 company.

Likewise, Barr, Boukhalfa and Bouibede (2018) used linear programming to solve the NP-hard problem of determining a horizontal fragmentation scheme in relational DW. Also, Nam, Kim and

Han (2018) proposed a graph-based database partitioning method called GPT that improves the performance of queries with less data redundancy. In Letrache, El Beggar and Ramdani (2018), it was proposed a dynamic fragmentation strategy for OLAP cubes, using association rule mining.

On the other hand, Kechar and Nait-Bahloul (2019) presented a horizontal data partitioning approach tailored to a large DW, interrogated through a high number of queries, the idea was to fragment horizontally only the large fact table based on partitioning predicates, elected from the set of selection predicates used by analytic queries. While, in Ramdane et al. (2019), the authors assured that horizontal partitioning techniques have been used for many purposes in big data processing, such as load balancing, skipping unnecessary data loads, and guiding the physical design of a DW. Therefore, they proposed a new data placement strategy in the Apache Hadoop environment called Smart Data Warehouse Placement —SDWP—, which allows performing star join operation in only one Spark stage. The problem of partitioning and load balancing in a cluster of homogeneous nodes was investigated; experiments using the TPC-DS benchmark, showed that the proposed method enhances OLAP query performances in terms of execution time.

Likewise, in Ramdane et al. (2019), authors mixed a data-driven and a workload-driven model to create a new scheme for distributed big data warehouses over Hadoop, called "SkipSJoin." First, SkipSJoin builds horizontal fragments (buckets) of the fact and dimension tables of the DW using a hash-partitioning method, and distributes these buckets evenly over the nodes of the cluster. Then, it allows skipping the scanning of some unnecessary data blocks, by hash-partitioning some DW tables with frequent attributes of the filters. With experiments using the TPC-DS benchmark they showed that the proposal outperforms some approaches in terms of query execution time.

Finally, in Hilprecht, Carsten and Uwe (2019), it was introduced that commercial data analytics products such as Microsoft Azure SQL Data Warehouse or Amazon Redshift provide ready-to use-scale-out database solutions for OLAP-style workloads in the cloud. Whereas the provisioning of a database cluster is in general fully automated by cloud providers, customers still have to make important design decisions which were traditionally made by the database administrator such as selecting the partitioning schemes, therefore, the authors proposed a learned partitioning advisor for analytical OLAP-style workloads based on Deep Reinforcement Learning —DRL—. The leading idea was that a DRL agent learns its decisions based on experience by monitoring the rewards for different workloads and partitioning schemes. The evaluation showed that the approach was not only able to find the partitioning that outperform existing approaches for automated partitioning design but that it can also easily adjust to different deployments.

Table 1 provides an analysis of the horizontal fragmentation methods discussed above, in which their classification and the validation used are identified.

Table 1. Comparative table of works on horizontal fragmentation

Work	Classification	Validation
Costa et al. (2016)	Hierarchy-based	Spatial Data Warehouse Benchmark (Spadawan)
Ettaoufik and Ouzzif (2017)	Cost-based	Benchmark APB-1
Barkhordari and Niamanesh (2018)	Map-Reduce-based	Benchmark TPC-DS
Boissier and Kurzynski (2018)	Cost-based	Benchmarks TPC-C, TPC-CH (CH-benCHmark), data and workload of a SAP ERP system of a Global 2000 company.
Barr, Boukhalfa and Bouibede (2018)	Metaheuristic-based	Benchmark APB-1

Work	Classification	Validation
Nam, Kim and Han (2018)	Cost-based Graph-based	Benchmark TPC-DS, The Internet Movie DataBase (IMDB) y BioWarehouse
Letrache, El Beggar and Ramdani (2018)	Data mining-based	Benchmark TPC-DS
Kechar and Nait-Bahloul (2019)	Cost-based Predicates-based	SSB
Ramdane et al. (2019)	Hash-partitioning-based	TPC-DS benchmark using Scala language on a cluster of homogeneous nodes, a Hadoop-YARN platform, a Spark engine, and Hive.
Ramdane et al. (2019)	Hash-partitioning-based	TPC-DS benchmark
Hilprecht, Carsten and Uwe (2019)	Deep Reinforcement Learning-based	Different databases schemata and workloads of varying complexity.

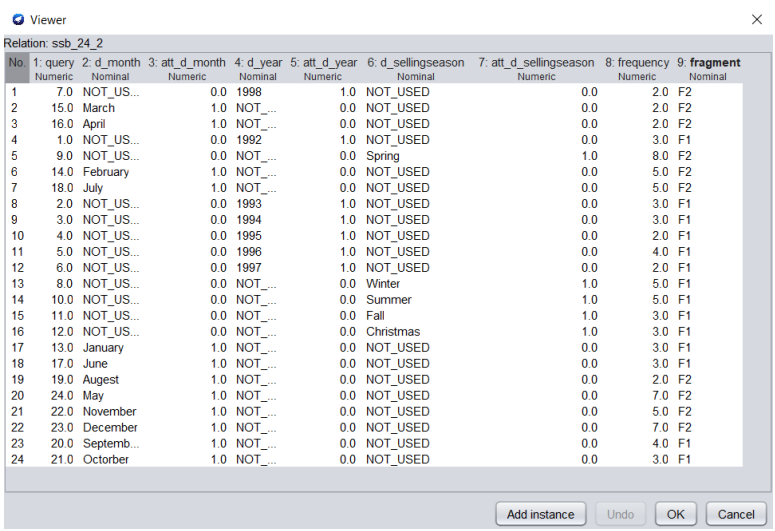
Source: author own elaboration.

Methodology

In this section, the process followed for the analysis of the decision tree algorithms is established; after that, each of the algorithms available in the version of Weka used are described.

Collection and Preparation of Data

In order to carry out the study of decision tree algorithms to select the best one to fragment the DW, we use SSB and PostgreSQL. We constructed eight data sets, the first four considering 24 queries and from two to five fragments, and the second four with 50 queries also from two to five fragments. We use the algorithm proposed by Rodríguez et al. (2014) to build the data sets. The resulting data set for 24 queries and two fragments is visualized in Figure 1.


Viewer

Relation: ssb_24_2

No.	1: query Numeric	2: d_month Nominal	3: att_d_month Numeric	4: d_year Nominal	5: att_d_year Numeric	6: d_sellingseason Nominal	7: att_d_sellingseason Numeric	8: frequency Numeric	9: **fragment** Nominal
1	7.0	NOT_US...	0.0	1998	1.0	NOT_USED	0.0	2.0	F2
2	15.0	March	1.0	NOT_...	0.0	NOT_USED	0.0	2.0	F2
3	16.0	April	1.0	NOT_...	0.0	NOT_USED	0.0	2.0	F2
4	1.0	NOT_US...	0.0	1992	1.0	NOT_USED	0.0	3.0	F1
5	9.0	NOT_US...	0.0	NOT_...	0.0	Spring	1.0	8.0	F2
6	14.0	February	1.0	NOT_...	0.0	NOT_USED	0.0	5.0	F2
7	18.0	July	1.0	NOT_...	0.0	NOT_USED	0.0	5.0	F2
8	2.0	NOT_US...	0.0	1993	1.0	NOT_USED	0.0	3.0	F1
9	3.0	NOT_US...	0.0	1994	1.0	NOT_USED	0.0	3.0	F1
10	4.0	NOT_US...	0.0	1995	1.0	NOT_USED	0.0	2.0	F1
11	5.0	NOT_US...	0.0	1996	1.0	NOT_USED	0.0	4.0	F1
12	6.0	NOT_US...	0.0	1997	1.0	NOT_USED	0.0	2.0	F1
13	8.0	NOT_US...	0.0	NOT_...	0.0	Winter	1.0	5.0	F1
14	10.0	NOT_US...	0.0	NOT_...	0.0	Summer	1.0	5.0	F1
15	11.0	NOT_US...	0.0	NOT_...	0.0	Fall	1.0	3.0	F1
16	12.0	NOT_US...	0.0	NOT_...	0.0	Christmas	1.0	3.0	F1
17	13.0	January	1.0	NOT_...	0.0	NOT_USED	0.0	3.0	F1
18	17.0	June	1.0	NOT_...	0.0	NOT_USED	0.0	3.0	F1
19	19.0	Augest	1.0	NOT_...	0.0	NOT_USED	0.0	2.0	F2
20	24.0	May	1.0	NOT_...	0.0	NOT_USED	0.0	7.0	F2
21	22.0	November	1.0	NOT_...	0.0	NOT_USED	0.0	5.0	F2
22	23.0	December	1.0	NOT_...	0.0	NOT_USED	0.0	7.0	F2
23	20.0	Septemb...	1.0	NOT_...	0.0	NOT_USED	0.0	4.0	F1
24	21.0	Octorber	1.0	NOT_...	0.0	NOT_USED	0.0	3.0	F1

Add instance | Undo | OK | Cancel

Figure 1. Data set with 24 queries and 2 fragments. Source: author own elaboration.

Application of Decision Tree Algorithms

The seven decision tree algorithms that offer the version of Weka 3.9.4 were applied to the eight data sets. A description of the algorithms is presented below.

Hoeffding Tree: It is an incremental, anytime decision tree induction algorithm that is capable of learning from massive data streams, assuming that the distribution generating examples does not change over time. Hoeffding trees exploit the fact that a small sample can often be enough to choose an optimal splitting attribute. This idea is supported mathematically by the Hoeffding bound, which quantifies the number of observations needed to estimate some statistics within a prescribed precision (Hulten, Spencer and Domingos, 2001).

Logistic Model Tree: Classifier for building LMT, which are classification trees with logistic regression functions at the leaves. The algorithm can deal with binary and multi-class target variables, numeric and nominal attributes and missing values (Landwehr, Hall and Frank, 2005).

J48: C4.5 Decision Tree is one of the most broadly used and real world approaches. In C4.5 the learned classifier is represented by a decision tree as sets of if-then rules to human readability improvement. The decision tree is simple to be understood and interpreted; besides, it can handle nominal and categorical data and perform well with large data set in short time. In C4.5 training, the decision tree is built in a top-down recursive way (Saeh et al., 2016).

Decision Stump: It is one level decision tree, that classifies instances by sorting them based on feature values. In a decision stump, each node represents a feature in an instance to be classified and each branch represents a value that the node can take. Instances are classified starting at the root node and sorting them based on their feature values (Kotsiantis, Tsekouras and Pintelas, 2005; Shi et al., 2018).

Random Forest: This algorithm uses bootstrap methods to create an ensemble of trees, one for each bootstrap sample. Additionally, the variables eligible to be used in splitting is randomly varied in order to decorrelate the variables. Once the forest of trees is created, they vote to determine the predicted value of input data (Dean, 2014).

Random Tree: It constructs a tree that considers a given number of random features at each node (Witten, Frank and Hall, 2011).

REPTree: It builds a decision or regression tree using information gain-variance reduction and prunes it using reduced-error pruning. Optimized for speed, it only sorts values for numeric attributes once. It deals with missing values by splitting instances into pieces, as C4.5 does. It can be set the minimum proportion of training set variance for a split, and number of folds pruning (Witten, Frank and Hall, 2011).

Results

After having analyzed the different decision tree algorithms, the following results were found for the Area ROC, Precision, Recall and F-measure metrics. Figure 2 to Figure 5 demonstrate that considering Recall, Precision, ROC Area and F-Measure metrics, respectively, for the 24 queries data sets, J48 algorithm was better for three, four and five fragments, only for two fragments it was overcome by Random Forest.

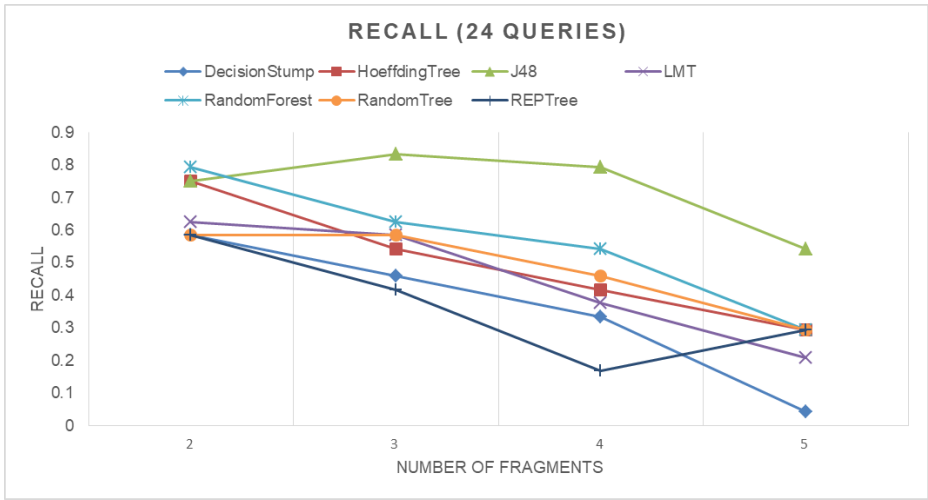


Figure 2. Results of Recall metric for 24 queries data sets. Source: author own elaboration.

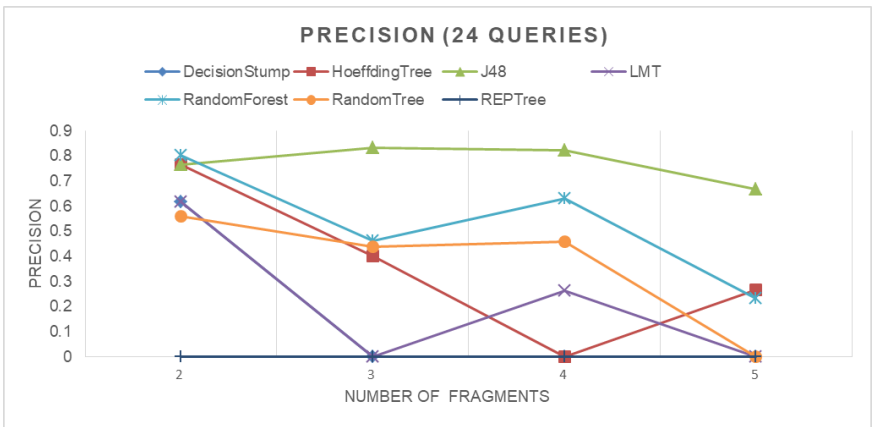


Figure 3. Results of Precision metric for 24 queries data sets. Source: author own elaboration.

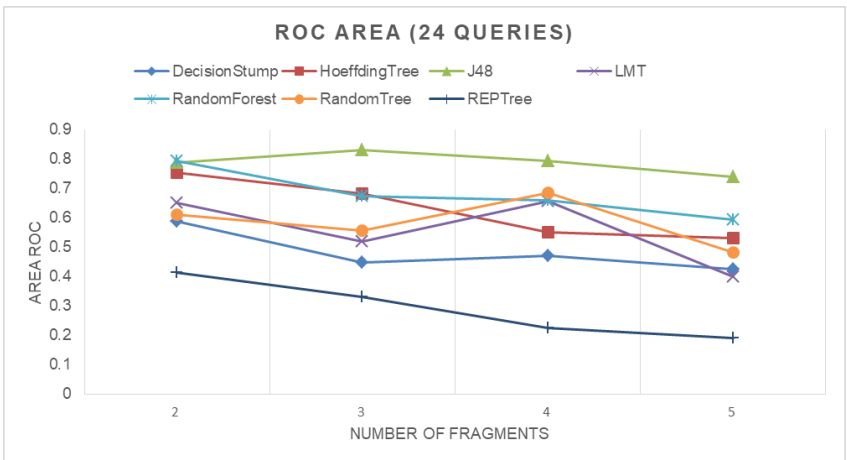


Figure 4. Results of ROC Area metric for 24 queries dataset. Source: author own elaboration.

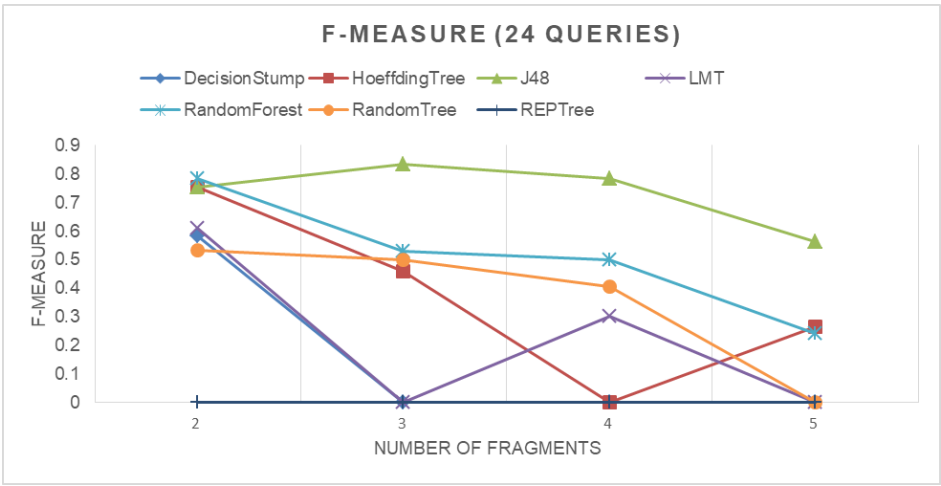


Figure 5. Results of F-Measure metric for 24 queries dataset. Source: author own elaboration.

With regards to the data sets of 50 queries, the results of the application of the decision tree algorithms presented in the Table 2 showed that for 2 fragments the best algorithm was REPTree because has a better behavior for the 4 metrics. While, the Table 3 demonstrates that for 3 fragments the best algorithm was Random Forest since it presented a better performance than the others. In the Table 4 the results for 4 fragments are shown, J48 was the best for major of metrics. Finally, in the Table 5 the best decision tree algorithm was Random Forest for five fragments.

Table 2. Results of decision trees algorithms with 50 queries for two fragments

Algorithm	Precision	Recall	ROC Area	F-Measure
Decision Stump	0.875	0.843	0.682	0.832
HoeffdingTree	0.875	0.843	0.885	0.832
J48	0.857	0.843	0.924	0.836
LMT	0.963	**0.961**	0.910	0.960
RandomForest	0.963	**0.961**	**0.998**	0.960
RandomTree	0.864	0.863	0.929	0.860
REPTree	**0.964**	**0.961**	0.994	**0.961**

Source: author own elaboration.

Table 3. Results of decision tree algorithms with 50 queries for three fragments

Algorithm	Precision	Recall	ROC Area	F-Measure
DecisionStump	0.561	0.686	0.691	0.617
HoeffdingTree	-	0.745	0.830	-
J48	0.681	0.725	0.679	0.693
LMT	0.722	0.725	0.907	0.723
RandomForest	**0.770**	**0.784**	**0.934**	**0.767**
RandomTree	0.654	0.647	0.782	0.627
REPTree	0.459	0.608	0.510	0.521

Source: author own elaboration.

Table 4. Results of decision tree algorithms with 50 queries for four fragments

Algorithm	Precision	Recall	ROC Area	F-Measure
DecisionStump	-	0.431	0.610	-
HoeffdingTree	0.500	0.353	0.645	0.353
J48	**0.709**	**0.706**	0.825	**0.707**
LMT	0.572	0.588	0.830	0.579
RandomForest	0.690	0.686	**0.886**	0.678
RandomTree	0.501	0.490	0.715	0.487
REPTree	0.548	0.490	0.701	0.487

Source: author own elaboration.

Once the analysis of the decision tree algorithms for 25 and 50 queries was concluded, it was determined that the two best algorithms were Random Forest and J48, so it was decided to select J48, since it is more efficient in building the model because the computational complexity of the J48 algorithm given set D is $O(n \times |D| \times \log(|D|))$, where n is the number of attributes describing the tuples in D and $|D|$ is the number of training tuples in D (Han, Kamber and Pei, 2012). In contrast, the time complexity for building forest of M randomized trees is $O(M \times K \times \tilde{N}^2 \times \log(\tilde{N}))$, where K is the number of variables randomly drawn at each node and $\tilde{N}$=0.632$|D|$ (Louppe, 2015). Figure 6 represents a decision tree created by J48 for the 50 queries data set and four fragments.

Table 5. Results of decision trees algorithms with 50 queries for five fragments

Algorithm	Precision	Recall	ROC Area	F-Measure
DecisionStump	-	0.294	0.591	-
HoeffdingTree	0.464	0.314	0.654	0.304

Algorithm	Precision	Recall	ROC Area	F-Measure
J48	0.671	0.647	0.834	0.642
LMT	0.657	0.667	0.895	0.661
RandomForest	**0.749**	**0.745**	**0.927**	**0.743**
RandomTree	0.610	0.569	0.779	0.566
REPTree	0.613	0.569	0.770	0.582

Source: author own elaboration.

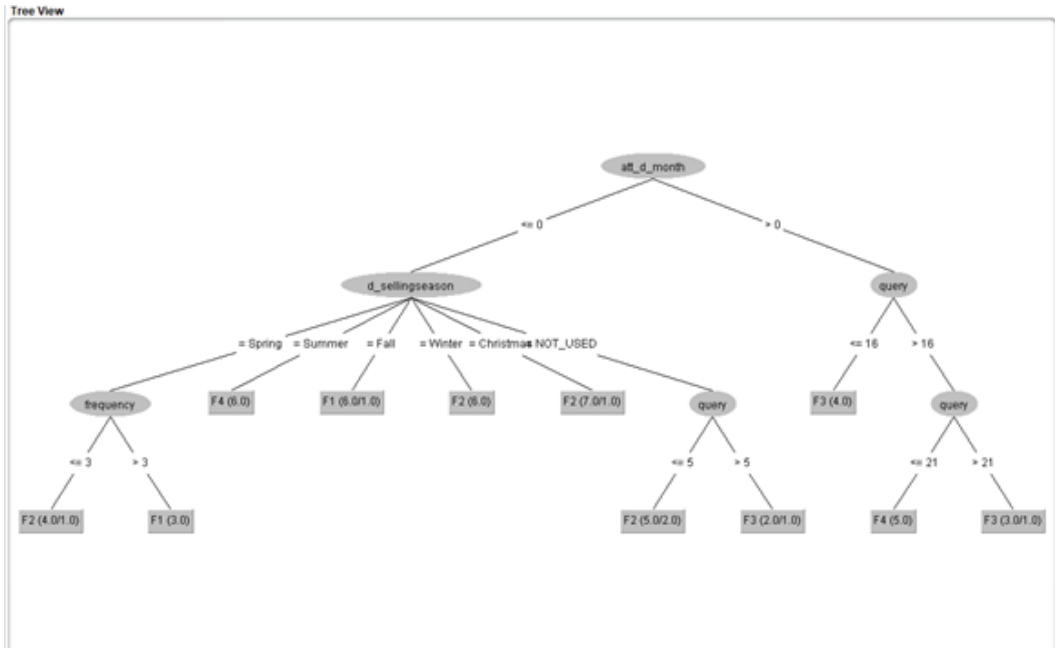


Figure 6. Decision tree created by J48. Source: author own elaboration.

Conclusions

DW are applied in several areas and allow efficient data analysis. Fragmentation in a DW allows optimizing response times and execution costs for OLAP queries. In this work it is proposed to take advantage of the potential of the decision trees in the classification to adapt them in the process of horizontal fragmentation of the DW for that reason, this article described the process in which the analysis of different decision trees algorithms was carried out, in order to determine the best of them to be implemented in a horizontal fragmentation method for data warehouses. As a result of the analysis, both J48 and Random Forest were the best algorithms for decision tree induction, and J48 was the selected algorithm for the method implementation because it has a time complexity lower than Random Forest. The future work is the design of the fragmentation method, which will consist of determining the most frequent OLAP queries, analyzing the predicates used by the queries, and based on this build the decision tree, from which the horizontal fragments will be generated. The method will be implemented in a Tourist Data Warehouse which is being implemented with data from official sources that regulate tourist activity in Mexico.

References

Barkhordari, M. and Niamanesh, M. (2018). Chabok: A Map-Reduce based method to solve data warehouse problems. *Journal of Big Data*, *5*(40), 1-25.

Barr, M., Boukhalfa, K. and Bouibede, K. (2018). Bi-Objective Optimization Method for Horizontal Fragmentation Problem in Relational Data Warehouses as a Linear Programming Problem. *Applied Artificial Intelligence*, *32*(9-10), 907-923.

Boissier, M. and Kurzynski, D. (2018). Workload-Driven Horizontal Partitioning and Pruning for Large HTAP Systems. In IEEE 34th International Conference on Data Engineering Workshops (ICDEW), Paris, France.

Costa, M.R. et al. (2016). Spatial data warehouses and spatial OLAP come towards the cloud: Design and performance. *Distributed and Parallel Databases*, *34*(3), 425-461.

Dean, J. (2014). *Big Data, Data Mining, and Machine Learning Value Creation for Business Leaders and Practitioners*. New Jersey, USA: John Wiley & Sons.

Ettaoufik, A. and Ouzzif, M. (2017). Web Service for Incremental and Automatic Data Warehouses Fragmentation. *International Journal of Advanced Computer Science and Applications*, *8*(6), 1-10.

Han, J., Kamber, M. and Pei, J. (2012). *Data Mining Concepts and Techniques*. Burlington, USA: Morgan Kaufmann Publishers.

Hilprecht, B., Carsten, B. and Uwe, R. (2019). *Learning a Partitioning Advisor with Deep Reinforcement Learning*. Recovered from https://arxiv.org/pdf/1904.01279.pdf.

Hulten, G., Spencer, L. and Domingos, P. (2001). Mining time-changing data streams. In Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

Kechar, M. and Nait-Bahloul, S. (2019). Bringing Together Physical Design and Fast Querying of Large Data Warehouses: A New Data Partitioning Strategy. In BDIoT'19: Proceedings of the 4th International Conference on Big Data and Internet of Things, Rabat, Morocco.

Kotsiantis, S., Tsekouras, G. and Pintelas, P. (2005). Local Bagging of Decision Stumps. In Ali, M. and Esposito, F. (Eds.), *Innovations in Applied Artificial Intelligence* (pp. 377-391). Berlin, Germany: Springer.

Landwehr, N., Hall, M. and Frank, E. (2005). Logistic Model Trees. *Machine Learning*, *59*(1-2), 161-205.

Letrache, K., El Beggar, O. and Ramdani, M. (2019). OLAP cube partitioning based on association rules method. *Applied Intelligence*, *49*(2), 420-434.

Louppe, G. (2015). *Understanding Random Forests: From Theory to Practice*. Liège, Belgium: Universidad of Liège.

Nam, Y.-M., Kim, M.-S. and Han, D. (2018). A Graph-Based Database Partitioning Method for Parallel OLAP Query Processing. In IEEE 34th International Conference on Data Engineering (ICDE), Paris, France.

Ozsu, M.T. and Valduriez, P. (2020). *Principles of Distributed Database Systems*. Geneva, Switzerland: Springer Nature Switzerland AG.

Ramdane, Y. et al. (2019). SDWP: A New Data Placement Strategy for Distributed Big Data Warehouses in Hadoop. In Ordonez, C. et al. (Eds.), *Big Data Analytics and Knowledge Discovery* (pp. 189-205). Berlin, Germany: Springer.

Ramdane, Y. et al. (2019). SkipSJoin: A New Physical Design for Distributed Big Data Warehouses in Hadoop. In Laender, A.H.F. et al. (Eds.), *Conceptual Modeling* (pp. 255-263). Berlin, Germany: Springer.

Rodríguez, L. et al. (2014). Horizontal Partitioning of Multimedia Databases Using Hierarchical Agglomerative Clustering. In Gelbukh, A. et al. (Eds.), *Nature-Inspired Computation and Machine Learning* (pp. 296-309). Cham, Switzerland: Springer.

Saeh, I.S. et al. (2016). Static Security classification and Evaluation classifier design in electric power grid with presence of PV power plants using C-4.5. *Renewable and Sustainable Energy Reviews*, *56*, 283-290.

Shi, L. et al. (2018). Signal prediction based on boosting and decision stump. *International Journal of Computational Science and Engineering*, *16*(2), 117-122.

Witten, I.H., Frank, E. and Hall, M. (2011). *Data Mining Practical Machine Learning Tools and Techniques*. New York, USA: Elsevier.