Revista Facultad de Ingeniería
Journal Homepage: https://revistas.uptc.edu.co/index.php/ingenieria

Decision Tree Algorithm Moderately Coupled to PostgreSQL DBMS

Ricardo Timarán-Pereira[1]
Anivar Chaves-Torres[2]
Hugo Ordoñez-Erazo[3]

Received: July 30, 2023 **Accepted:** October 08, 2023 **Published:** November 21, 2023

Citation: R. Timarán-Pereira, A. Chaves-Torres, H. Ordoñez-Erazo, "Decision Tree Algorithm Moderately Coupled to PostgreSQL DBMS," *Revista Facultad de Ingeniería*, vol. 32, no. 66, e16777, 2023. https://doi.org/10.19053/01211129.v32.n66.2023.16777

Abstract

Using machine learning for data management is an extraordinary opportunity to move towards a leadership model based on information, which drives the organization towards success in each initiative. However, when incorporating these technologies, a company presents problems associated with the economic and administrative costs generated in this process since these are usually quite high, limiting their implementation in MSMEs. This paper proposes to integrate supervised machine learning techniques into PostgreSQL DBMS in a moderately coupled

[1] Ph. D. Universidad de Nariño (Pasto-Nariño, Colombia). ritimar@udenar.edu.co. ORCID: 0000-0002-0006-6654
[2] Ph. D. Universidad de Nariño (Pasto-Nariño, Colombia). anivar.chaves@unad.edu.co. ORCID: 0000-0001-6984-7398
[3] Ph. D. Universidad del Cauca (Popayán-Cauca, Colombia). hugoordonez@unicauca.edu.co. ORCID: 0000-0002-3465-5617

architecture to provide it with the capabilities of discovering knowledge in databases. Classification and regression algorithms were coupled by developing extensions using one of the procedural languages supported by PostgreSQL. Initially, the C4.5 decision tree classification algorithm was implemented using the PL/pgSQL procedural language. The main advantage of this strategy is that it considers the scalability, administration, and data manipulation of the DBMS. Since PostgreSQL is an open-source manager, organizations such as MSMEs will have a free tool that allows them to perform predictive analysis in order to improve their decision-making processes by anticipating future consumer behavior and making rational decisions based on their findings.

Keywords: classification techniques; C4.5 algorithm; moderately coupled architecture; PostgreSQL.

Algoritmo de árboles de decisión medianamente acoplado a PostgreSQL

Resumen

El uso de Aprendizaje Automático para la gestión de datos es una oportunidad extraordinaria para avanzar hacia un modelo de liderazgo basado en la información, que impulse a la organización hacia el éxito en cada una de sus iniciativas. Sin embargo, una empresa, en el momento de incorporar estas tecnologías presenta problemáticas asociadas con los costos económicos y administrativos generados en este proceso, ya que estos suelen ser bastante elevados, que limita principalmente a las MiPymes, su implementación. En este artículo se presenta la propuesta de integrar al SGBD PostgreSQL, técnicas supervisadas de aprendizaje automático, en una arquitectura medianamente acoplada, con el fin de dotar a este gestor con las capacidades de descubrir conocimiento en las bases de datos. Se acoplarán algoritmos de clasificación y regresión mediante el desarrollo de extensiones utilizando uno de los lenguajes procedurales soportados por PostgreSQL. Inicialmente, se implementará el algoritmo de clasificación por árboles de decisión C4.5 usando el lenguaje procedural PL/pgSQL. La principal ventaja de esta estrategia es que se tiene en cuenta la escalabilidad, administración y manipulación de datos del SGBD. Al ser PostgreSQL un gestor de código abierto,

organizaciones tales como MiPymes, contarán con una herramienta libre que les permita realizar análisis predictivo con el fin mejorar sus procesos de toma de decisiones al poder anticiparse a los futuros comportamientos del consumidor y tomar decisiones racionales basadas en sus hallazgos.

Palabras clave: algoritmo C4.5; arquitectura medianamente acoplada; PostgreSQL; técnicas de clasificación.

Algoritmo de árvore de decisão fracamente acoplado ao PostgreSQL

Resumo

A utilização de Machine Learning para gestão de dados é uma oportunidade extraordinária para avançar para um modelo de liderança baseado em informação, que impulsiona a organização para o sucesso em cada uma das suas iniciativas. No entanto, uma empresa, ao incorporar estas tecnologias, apresenta problemas associados aos custos económicos e administrativos gerados neste processo, uma vez que estes são normalmente bastante elevados, o que limita principalmente a sua implementação às MPME. Este artigo apresenta a proposta de integrar técnicas de aprendizado de máquina supervisionado ao SGBD PostgreSQL em uma arquitetura moderadamente acoplada, a fim de dotar este gestor de capacidades de descoberta de conhecimento em bancos de dados. Algoritmos de classificação e regressão serão acoplados através do desenvolvimento de extensões utilizando uma das linguagens procedurais suportadas pelo PostgreSQL. Inicialmente, o algoritmo de classificação de árvore de decisão C4.5 será implementado utilizando a linguagem processual PL/pgSQL. A principal vantagem desta estratégia é que a escalabilidade, administração e manipulação de dados do SGBD são levadas em consideração. Dado que o PostgreSQL é um gestor de código aberto, organizações como a MiPymes terão uma ferramenta gratuita que lhes permite realizar análises preditivas para melhorar os seus processos de tomada de decisão, sendo capazes de antecipar comportamentos futuros dos consumidores e tomar decisões racionais com base nas suas descobertas.

Palavras-chave: Algoritmo C4.5; arquitetura fracamente acoplada; PostgreSQL; técnicas de classificação.

I. INTRODUCTION

Machine learning is a field of study of artificial intelligence that, based on databases, launches algorithms to obtain predictive analyses for precise purposes and establishes correlations between various events. Machine learning algorithms find natural patterns in data that generate insights and help make better decisions and predictions. Predictive models are used to anticipate, predict, and make decisions. Using machine learning for data management is an extraordinary opportunity to move towards an information-based leadership model, which drives the organization towards success in each initiative.

However, when incorporating these technologies, an organization presents problems associated with the economic and administrative costs generated in the process since these are usually quite high because the companies offering these products have projects developed to satisfy each of the needs that may be found in any organization. Only large organizations can carry out the respective implementations due to their economic capacity. However, for micro, small, and medium-sized companies (MSMEs), adopting projects of these characteristics is unattainable, making them the most forgotten and ignored throughout the literature on data analysis and decision making. To the extent that it might seem that machine learning is an inhospitable terrain not suitable for small entrepreneurs, dominated by large industries and the most powerful multinationals.

DataBase Management Systems (DBMS) are the most important component of any data-intensive pattern discovery application. However, the drawback is that most DBMSs must carry out their knowledge discovery processes outside the database, in what Timarán [1] calls loosely coupled architecture with the DBMS using other tools, which sometimes require expensive licenses and are not available to companies such as MSMEs. On the other hand, in a loosely coupled architecture, the database manager is only used for storage, missing the fact that they have query optimization techniques that can improve query performance.

Instead, in a moderately coupled architecture (Figure 1), specific machine learning tasks and algorithms are implemented in the DBMS as stored procedures, user-defined functions, or extensions. Coupling through extensions is done through user-

defined functions (UDFs) defined and implemented in a general-purpose programming language. The executable of a UDF is stored in the DBMS, and it can access and invoke the function when it is referenced in an SQL statement. Machine learning algorithms are expressed as a collection of user-defined functions (UDFs). Most processing is done in the UDF, and the DBMS is primarily used to provide tuples to the UDFs. UDFs, like stored procedures, run in the same addressing space as the DBMS.

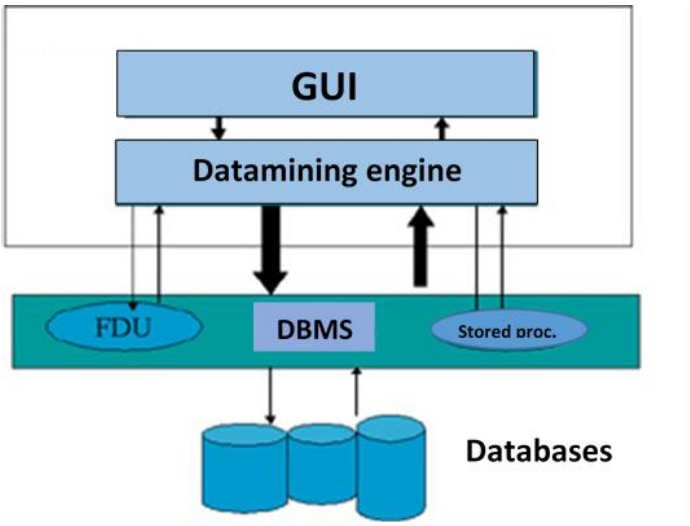

Fig. 1. Moderately coupled architecture with DBMS [1].

The main advantage of UDFs over stored procedures is their performance since passing tuples to a stored procedure is slower than to a UDF. The rest of the processing is the same. The main advantage of this strategy is that it considers the scalability, administration, and data manipulation of the DBMS. Its biggest challenge is to obtain good performance since the DBMS query optimizer does not have new search and transformation strategies that allow it to generate efficient execution plans for queries that involve the discovery of patterns or knowledge [1].

In the case of the PostgreSQL DBMS, one of the most used open-source database managers on the market, specific functionalities can be added through libraries called extensions to implement new processes or algorithms without leaving the database. Extensions are libraries that are added to PostgreSQL to add specific functionality. PostgreSQL allows user-defined functions to be written in languages

other than SQL, PL/pgSQL, and C [2]. Support for additional languages such as PL/Perl, PL/Python, PL/V8 (also known as PL/JavaScript), PL/Ruby, and PL/R can be easily enabled. This support for a wide variety of languages allows to choose the language with constructions that can better solve the problems [2].

Extensions are plugins that can be installed on a PostgreSQL database to extend functionality beyond the base offerings; starting with PostgreSQL 9.1+, plugins are easily installed using the new PostgreSQL extension model [2]. Several extensions and projects have been proposed that allow machine learning capabilities to be integrated into the PostgreSQL DBMS. For example, Hellerstein et al. [3] propose MADlib, a free open-source library for scalable analysis of databases. It provides data-parallel implementations of mathematical, statistical, graphical, and machine learning methods for structured and unstructured data. Moreover, it offers various SQL-based algorithms for machine learning, data mining, and statistics, supporting several relational databases, including PostgreSQL. These run at scale within a database engine without the need to import/export data to other tools.

Carrigan and Torres [4] propose MindsDB, an open-source tool that uses artificial intelligence to facilitate machine learning on databases, including PostgreSQL. It allows users to create machine learning models and make predictions directly within the database, simplifying the process of deploying and using machine learning in enterprise applications. The platform is designed to be easy to use and allows users, even those without machine learning experience, to harness the power of machine learning to make data-driven predictions and decisions.

Robles and Sotolongo [5] propose integrating algorithms from data mining techniques, induction rules, and decision trees into the PostgreSQL database management system through extensions. Among the advantages of this proposal is that, instead of executing an SQL script to load “separate” objects in your database, the extension will be a package containing all the objects defined in it, which brings excellent benefits when updating or deleting it. The implemented algorithms were validated through an experimental design that allowed us to observe that the analysis times of the algorithms integrated into the DBMS are shorter than the results of the Weka tool.

Castro, Cabrera and Timarán [6] propose Mate-KDD Data Mining Tool, an open-source desktop data mining tool, for the classification task that implements the C4.5 machine learning algorithms, the Mate-tree algorithm and SLIQ, through user-defined functions (UDFs) from data stored in a database in the PostgreSQL DBMS in a moderately coupled architecture. It offers support in all phases of the knowledge discovery process in databases (selection, preprocessing, transformation, data mining, and evaluation), in addition to operating directly within the DBMS by applying the algorithms implemented in UDFs, which means that its architecture is moderately coupled with the manager, this is favorable in the sense that it works directly with a database and applies the mining process from the manager, gaining processing speed. A graphical interface allows the user (analyst) to interact with the manager easily, and the classifier can be configured by selecting the algorithm to apply, defining the percentage of data partition for training and testing to apply the model and obtain results graphically by being able to visualize the tree and the rules generated in a detailed and precise way.

García-Tembleque [7] proposes the implementation of a tool for multivariate analysis within the context of databases and machine learning on regularized versions of PCA (Principal Component Analysis), CCA (Canonical Correlation Analysis), and OPLS (Orthonormalized PartialLeast Squares) via Apache Spark and Python that provides a general idea on how to implement this type of analysis to a moderately coupled DBMS.

Sotolongo [8] proposes a solution for migrating data from an SGDB to PostgreSQL to send emails, which is not within the functionalities of PostgreSQL. By using extensions, the functionality of this manager is expanded in a moderate coupling. Using plpython and the python smtp standard library, the author creates the pgsmtp extension and shares it with the community along with its source code.

PostPic is an open-source extension for PostgreSQL created by Rotiroti [9] that allows image processing within the database, as PostGIS does for spatial data. It adds the new “image” data type to SQL and several functions to process images and extract their attributes. It is currently located within the official PostgreSQL extensions repository.

Díaz [10] proposes an extension based on the R language to graph in PostgreSQL. In this research, version 3.0 of an extension for the PostgreSQL manager (pgr-graphic) was developed, which allows the graphical representation of data contained in databases using the R statistical package as a graphical engine. The use of the extension allows the manager to save and manage a large amount of data and provides an analysis and graphical output, eliminating external services.

This article proposes to expand the functionalities of the PostgreSQL DBMS through the implementation of extensions in the Pl/pgSQL procedural language, initially integrating the C4.5 decision tree classification algorithm in a moderately coupled architecture. Providing the PostgreSQL DBMS with these machine learning techniques gives organizations such as MSMEs a free tool to perform predictive analysis to improve their decision-making processes by anticipating future consumer behaviors and making rational decisions based on their findings.

II. Materials and Methods

Descriptive studies focus on and are designed to describe the distribution of variables without considering causal or other hypotheses. Thus, this research was descriptive under a quantitative approach, applying a non-experimental design.

Creating an extension in PL/pgSQL to integrate the C4.5 algorithm into PostgreSQL is an ambitious project that involves several steps and a good knowledge of both PostgreSQL and the C4.5 algorithm. The CRISP-DM (Cross-Industry Standard Process for Data Mining) methodology was adapted, as it is research that involves data mining techniques [11-13]. CRISP-DM comprises six phases: problem analysis, data analysis, data preparation, modeling, evaluation, and implementation.

In the problem analysis phase, the bibliography that allowed researchers to gain appropriate knowledge about creating extensions to provide the PostgreSQL DBMS with the capabilities of using classification techniques for knowledge discovery in a moderately coupled architecture was collected and selected.

This process allowed collecting the correct information to obtain adequate results. In the data analysis phase, the decision tree classification technique and the C4.5 algorithm were identified, collected, and familiarized. In the data preparation phase,

the different functions that must be created for the PostgreSQL extension with the classification technique were identified and designed. In the modeling phase, the extension called C4.5 was created, and the functions identified above were developed in the Pl/pgSQL procedural language and integrated into the PostgreSQL C4.5 extension.

In the evaluation phase, several data sets were selected, and the performance of the C4.5 extension was evaluated on these data sets. In the implementation phase, the entire research process, code, and instructions on installing and using the extension were documented. Moreover, the extension was prepared for distribution and download by interested MSMEs, who will be given the necessary training for its use and implementation.

III. Results

A. Implementation of Extensions

The classification model based on decision trees originates in machine learning studies. This is a supervised learning method that builds decision trees from a set of cases or examples called training set extracted from the database. A test set, whose characteristics are known, is also chosen in order to evaluate the tree. This model is probably the most used and popular due to its simplicity and ease of understanding [14-16]. The importance of decision trees is due to their ability to build interpretable models, this being a decisive factor for their application. Classification with decision trees considers disjoint classes so that the tree will lead to one and only one leaf, assigning a single class to the prediction [12]. Therefore, this technique was chosen to be implemented initially in this research.

The decision tree algorithm implemented was C4.5 [17]. This algorithm is based on the use of the gain ratio criterion. In this way, it is possible to prevent the variables with the greatest number of possible values from benefiting from the selection. The basic idea of this algorithm is to construct decision trees in which each non-terminal node is labeled with an attribute. Each branch leaving a node is labeled with a value of that attribute, and each terminal node is labeled with a set of cases, each of which satisfies all attribute values that label the path from that node to the starting node.

The classification power of an attribute is measured according to its ability to reduce uncertainty or entropy (degree of disorder in a system). This metric is called information gain. The gain ratio is calculated from this metric. The attribute with the highest gain ratio is chosen as the attribute that forms a node in the tree [10].

To integrate the C4.5 algorithm into PostgreSQL through the extension called decision_tree, the files decision_tree.control and decision_tree--1.0.sql were initially created in the path that PostgreSQL designated for the creation of extensions, which is: C:\Program Files\PostgreSQL\14\share\extension.

Once these files were created, the extension was created in the database using the SQL command: ***CREATE EXTENSION*** *decisión_tree;*

The decision_tree control file contains the decision_tree—1.0.sql file configuration, which contains the functions in the Pl/pgSQL procedural language that implements the C4.5 algorithm. The contents of the decision_tree control file shown in Figure 2, and the prototype functions of the decision_tree—1.0.sql file are shown in Figure 3.

```
# decision_tree.control
comment = 'extension clasificación por arboles de decision'
default_version = '1.0'
relocatable = false
schema = pg_catalog
superuser = true
trusted = true
```

Fig. 2. Content file decisión_tree.control.

```
– decision_tree--1.0.sql

CREATE OR REPLACE FUNCTION C45(varchar) RETURNS SETOF
varchar language 'plpgsql' AS $body$              $body$;

CREATE OR REPLACE FUNCTION c45_entro(integer,varchar) RETURNS
SETOF datentro  language 'plpgsql'  AS $body$              $body$;

CREATE OR REPLACE FUNCTION c45_gain(integer,float,varchar)
RETURNS SETOF datentro language 'plpgsql'  AS $body$              $body$;

CREATE OR REPLACE FUNCTION c45_tree(integer,integer,integer,varchar)
RETURNS SETOF integer language 'plpgsql'  AS $body$              $body$;
```

Fig. 3. Prototype functions in file decisión_tree—1.0.sql.

Figure 3 shows that the c45() function receives as a parameter a VARCHAR data type, which is the name of the class attribute of the data set. This is the main function of the algorithm, and it works with a “kdd_ent” table where the training data must be properly transformed. It also requires the subfunctions c45_entro(), c45_gain(), and c45_tree(). First, this function is responsible for physically creating the tables c45_rules and c45_nodes as the auxiliary tables used during the modeling process. It then calls the function c45_entro (), which returns the entropy of the class and continues searching for the attribute that will be used for the root node using the function c45_gain (). Finally, it calls the function c45_tree (), which finishes creating the classification model. Both this function and c45_tree () use a temporary table called pg_columna that acts as a vector and keeps the order and number of the attributes that make up a path of the tree, that is, a branch. Figure 4 shows the relationships of the functions necessary to develop the C4.5 algorithm.

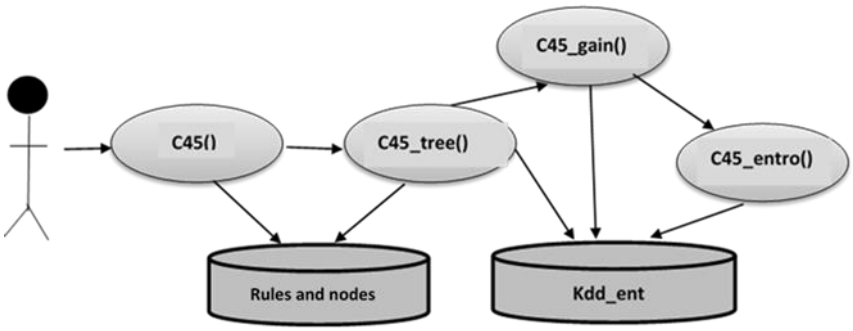

Fig. 4. C4.5 algorithm implementation.

The c45_tree() function receives an INTEGER data type as a parameter containing the number of attributes that have made up a condition so far. This function is called by the main function, c45(), or by itself. It starts by inserting the current node into the c45_nodes table and then validates that this node is a terminal leaf. If so, it builds the rules by going through the auxiliary table pg_columna and inserts it into the c45_rules table. If it is not a terminal node, the analysis continues, first from the left and then from the right, taking the attributes that have not been used and the attribute

with the greatest gain. To obtain the information gain, it uses the function c45_gain (). The next node in the tree then calls the c45_tree () function again.

The c45_entro () function receives as a parameter an INTEGER data type that indicates whether the entropy of the class or the entropy of a specific group of attributes is required, and together with a VARCHAR data type that contains the name of the class attribute. This function calculates the entropy by applying Shannon's entropy formula, for the C4.5 algorithm, and returns the entropy value, which is a NUMERIC data type.

The c45_gain () function receives a FLOAT data type as a parameter containing an entropy value, the data necessary to obtain the gain, along with a VARCHAR data type that contains the name of the class attribute of the table. This function calculates the gain of an attribute using Shannon's entropy () function for algorithm C4.5.

B. Functionality Tests

To test the functionality of the decision_tree extension implemented in PostgreSQL, the typical data set called playing tennis was taken (table 1). The objective is to discover patterns that allow us to determine whether tennis is played or not (class attribute or dependent variable), depending on the values of the attributes sky, temperature, humidity and wind (predictive attributes or independent variables) in fourteen days.

Table 1. Data set play tennis.

Day	Sky	Temperature	Humidity	Wind	Play
D1	Sunny	85	85	Weak	No
D2	Sunny	80	90	Strong	No
D3	Overcast	83	78	Weak	Yes
D4	Rain	70	96	Weak	Yes
D5	Rain	68	80	Weak	Yes
D6	Rain	65	70	Strong	No
D7	Overcast	64	65	Strong	Yes
D8	Sunny	72	95	Weak	No
D9	Sunny	69	70	Weak	Yes
D10	Rain	75	80	Weak	Yes
D11	Sunny	75	70	Strong	Yes
D12	Overcast	72	90	Strong	Yes
D13	Overcast	81	75	Weak	Yes
D14	Rain	71	80	Strong	No

When executing the C45() function using the select c45('play_tennis') command, this set of data is transformed into the format needed by the c45() function and is internally renamed kdd_ent, which contains only numerical values and organizes the attributes in such a way that the class attribute is at the end. Table 2 shows the result.

Table 2. Transformed play tennis dataset.

Sky	Temperature	Humidity	Wind	Play
0	2	1	0	0
0	2	2	1	0
2	2	1	0	1
1	0	2	0	1
1	0	1	0	1
1	0	0	1	0
2	0	0	1	1
0	1	2	0	0
0	0	0	0	1
1	1	1	0	1
0	1	0	1	1
2	1	2	1	1
2	2	0	0	1
2	1	1	1	0

Additionally, executing the c45() function generates two tables, c45_rules (Table 3) and c45_nodes (Table 4). The c45_rules table presents the related nodes in the same rule sequentially. All records that make up a rule have the same value in the id attribute, which indicates the rule number. The c45_nodes table specifies the relationships between the nodes in the tree and the useful values. All values are transformed and, thus, can be translated into real and coherent results.

Table 3. C45_rules table.

Id	Attribute	Values
1	1	0
1	3	0
1	5	1
2	1	0
2	3	1
2	5	0
3	1	1
3	4	0
3	5	1
4	1	1
4	4	1
4	5	0

Id	Attribute	Values
5	1	2
5	5	1

Table 4. C45_nodes table**.**

Node	Father	Attribute	Value	Son	Hemi	Class
0	-1	1	-1	-1	-1	-1
1	0	1	0	3	2	-1
2	0	1	1	5	-1	-1
3	1	3	0	-1	4	1
4	1	3	1	-1	-1	0
5	4	4	0	-1	6	1
6	4	4	1	-1	7	0
7	0	1	2	-1	-1	1

The interpretation of the rules in Table 3 is shown in Table 5, and the nodes in Table 4 are shown in Table 6.

Table 5. C45_rules interpreted table**.**

Id	Attribute	Value
1	Sky	Sunny
1	Humidity	<= 80.5
1	Play	Yes
2	Sky	Sunny
2	Humidity	>80.5
2	Play	No
3	Sky	Rain
3	Wind	Weak
3	Play	Yes
4	Sky	Rain
4	Wind	Strong
4	Play	No
5	Sky	Overcast
5	Play	Yes

Table 6. C45_nodes interpreted table.

Node	Father	Attribute	Value	Class
0	-1	Sky	-1	-1
1	0	Sky	Sunny	-1
2	1	Humidity	<=80.5	Yes
3	1	Humidity	>80.5	No
4	0	Sky	Rain	-1
5	4	Wind	Weak	Yes
6	4	Wind	Strong	No
7	0	Sky	Overcast	Yes

To generate the rules, the arm_rules() function was defined, which receives a VARCHAR data type as a parameter containing the name of the class attribute and a VARCHAR data type containing the name of the classification algorithm, which in this case is c45. If the arm_rules() function is applied to the rule table c45_rules (Table 5), the results are the rules shown in Figure 5.

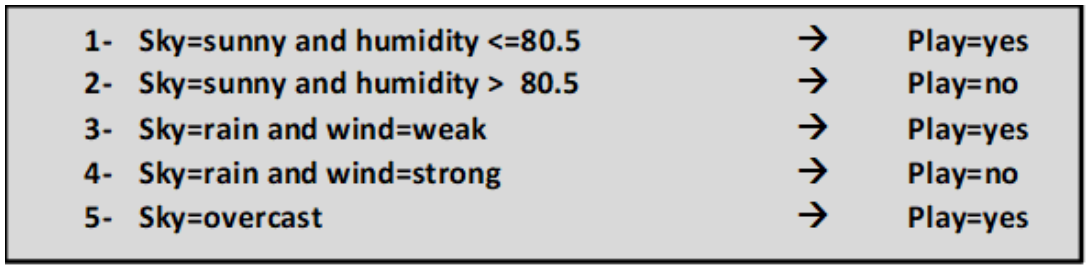

Fig. 5. Rule generation with C4.5 algorithm.

IV. Conclusions

The integration process of the C4.5 algorithm of the supervised decision tree classification technique was presented by creating an extension called decision_tree written in the Pl/pgSQL language, to the PostgreSQL database manager, in a moderately coupled architecture. The results of the functionality tests carried out demonstrated that this extension works correctly. This is the initial step in implementing new machine learning techniques as extensions in the PostgreSQL DBMS.

Future work includes testing the functionality of the decision_tree extension with real data sets and evaluating its performance. On the other hand, there is the implementation of new supervised (predictive) and descriptive (unsupervised) machine learning techniques, such as extensions in the PostgreSQL DBMS, as well as a graphical environment to improve and facilitate the usability of these techniques for the end user. This will allow organizations such as MSMEs to have a free tool to perform predictive analysis to improve their decision-making processes by anticipating future consumer behaviors and making rational decisions based on their findings.

AUTHORS' CONTRIBUTION

Ricardo Timarán-Pereira: research, software, writing-edit and review.

Anivar Chaves-Torres: research, conceptualization, writing-original draft.

Hugo Ordoñez-Erazo: research, writing-edit and review.

REFERENCES

[1] R. Timarán, "Arquitecturas de Integración del Proceso de Descubrimiento de Bases de Datos con Sistemas de Gestión de Bases de Datos," *Revista Ingeniería y Competitividad*, vol. 3, no. 2, pp. 45-55, 2001. https://doi.org/10.25100/iyc.v3i2.2327

[2] PostgreSQL Global Development Group, *PostgreSQL 15.3 Documentation*, 2023. https://www.postgresql.org/files/documentation/pdf/15/postgresql-15-US.pdf

[3] J. M. Hellerstein *et al.*, "The MADlib analytics library or mad skills, the SQL," *Proceedings of the VLDB Endowment*, vol. 5, no. 12, pp. 1700–1711. 2012. https://doi.org/10.14778/2367502.2367510

[4] A. Carrigan, J. Torres, *MinsDB*, 2023. https://docs.mindsdb.com/what-is-mindsdb

[5] Y. Robles, A. Sotolongo, "Integración de los algoritmos de minería de datos 1R, PRISM e ID3 a PostgrSQL," *Revista de Gestão da Tecnologia e Sistemas de Informação*, vol. 10, no. 2, pp. 389-406, 2014. https://doi.org/10.4301/S1807-17752013000200012

[6] C. Castro, M. Cabrera, R. Timarán, *MateKDD: una herramienta de minería de datos medianamente acoplada con PostgreSQL*, 2023. http://grias.udenar.edu.co/grias/?p=239

[7] A. García-Tembleque, "Implementación de Algoritmos de Aprendizaje Automático para Big Data," Grade Thesis, Universidad Carlos III, Spain, 2017. https://e-archivo.uc3m.es/handle/10016/27534

[8] A. Sotolongo, *pgsmtp: enviando correos desde PostgreSQL*, 2018. https://anthonysotolongo.wordpress.com/2018/05/28/pgsmtp-enviando-correos-desde-postgresql/

[9] D. Rotiroti, *PostPic: A PostgreSQL extensión for image-processing*, 2023. https://github.com/drotiro/postpic

[10] C. Díaz, "Extensión basada en R para graficar en PostgreSQL," Grade Thesis, Universidad de las Ciencias Informáticas, Cuba, 2014. https://repositorio.uci.cu/jspui/bitstream/ident/9246/2/TD_07692_14.pdf

[11] A. Azevedo, M. Santos, "KDD, SEMMA and CRISP-DM: a parallel overview," in *Proceedings of IADIS European Conference on Data Mining*, 2008, pp. 182-185.

[12] J. Hernández, M. Ramirez, C. Ferri, *Introducción a la Minería de Datos*, Editorial Pearson Prentice Hall, Spain, 2005.

[13] J. Villena, *CRISP-DM: La metodología para poner orden en los proyectos de Data Science*, 2016. https://data.sngular.team/es/art/25/crisp-dm-la-metodologia-para-poner-orden-en-los-proyectos-de-data-science

[14] J. Han, M. Kamber, J. Pei, *Data Mining: Concepts and Techniques*, Third Edition, Burlington, MA: Morgan Kaufmann, 2011.

[15] R. Timarán, M. Millán, "New algebraic operators and SQL primitives for mining classification rules", in *Computational Intelligence*, USA, 2006, pp. 61–65.

[16] K. Sattler, O. Dunemann, "SQL database primitives for decision tree classifiers", in *Proceedings of the tenth international conference on Information and knowledge management*, 2001, pp. 379–386.

[17] J. R. Quinlan, *C 4. 5: Programs for Machine Learning*, Morgan Kaufmann Publishers. San Francisco, 1993.