Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 10, No. 4, Mes Abril, 2017
ISSN: 2306-2495 | RNPS: 2343
Pág. 16-28
http://publicaciones.uci.cu

Tipo de artículo: Artículo original
Temática: inteligencia artificial
Recibido: 22/02/17 | Aceptado: 20/03/17 | Publicado: 24/04/17

Sistema para la generación de reglas de clasificación utilizando algoritmo de programación genético

System for the generation of classification rules using genetic programming algorithm

José Leandro González González[1], Omar Mar Cornelio[2], Pedro M. Puid Díaz [3]

[1] Facultad de Ciencias y Tecnologías Computacionales, Universidad de las Ciencias Informáticas. jlgonzalez@uci.cu
[2] Facultad de Ciencias y Tecnologías Computacionales, Universidad de las Ciencias Informáticas. omarmar@uci.cu
[3] Facultad de Ciencias y Tecnologías Computacionales, Universidad de las Ciencias Informáticas. pmpuig@uci.cu

[*] Autor para correspondencia: omarmar@uci.cu

Resumen

La extracción de conocimiento representa la base sobre la toma de decisiones en diversos procesos, siendo la minería de datos un área del conocimiento fundamental para garantizar dicho objetivo. Dentro de los procesos más utilizados en la actualidad se encuentran la Inteligencia Artificial y el Análisis Estadístico implementado mediante técnicas de Asociación, Agrupamiento, Clasificación que son desarrolladas mediante métodos estocásticos, heurística, redes neuronales. Sin embargo no se cuenta con herramientas automatizadas que garantice el análisis de procesos biológicos que permita su generalización. La presente investigación describe una solución a la problemática planteada a partir de la codificación de un sistema informático que implementa algoritmo de programación genético para la elaboración de reglas de clasificación

Palabras clave: reglas de clasificación; algoritmo genético; programación genética.

Abstract

The extraction of knowledge is the basis for decision making in various processes, data mining remains an area of critical knowledge to attain that end. Among the most currently used processes are Artificial Intelligence and Statistical Analysis techniques implemented by Association, grouping, classification that are developed by stochastic heuristics, neural networks methods. But you do not have automated tools that ensures analysis of biological processes that allow generalization. This paper describes a solution to the issues raised from the coding of a computer system implementing genetic programming algorithm for the development of classification rules.

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 10, No. 4, Mes Abril, 2017
ISSN: 2306-2495 | RNPS: 2343
Pág. 16-28
http://publicaciones.uci.cu

Keywords: *classification rules; genetic algorithm; genetic programming.*

Introducción

La Minería de Datos es actualmente un tema muy difundido, el cual engloba una serie de procesos y técnicas muy novedosas basadas en la Inteligencia Artificial y el Análisis Estadístico, encaminadas a la extracción de "conocimiento" procesable, nuevo y útil, implícito en grandes almacenes de datos y/o bases de datos (Pistohl, Joshi, Ganesh, Jackson, & Nazarpour, 2015). Para encontrar este conocimiento implícito en la información, esta rama se apoya en la realización de diferentes tareas, en dependencia del problema que se plantee, tales como:

Asociación, Agrupamiento (clustering en ingles), Clasificación, entre otras, siendo esta última una de las más usadas. Como métodos de solución a estas tareas se encuentran los algoritmos estadísticos, redes neuronales, Algoritmos Evolutivos o algoritmos de búsqueda basados en reglas probabilísticas, entre otros, y estos últimos presentan variantes como los algoritmos genéticos y la programación genética. Dada la efectividad de las tareas antes mencionadas, se sugiere su aplicación en herramientas que apoyen las investigaciones realizadas en diferentes áreas científicas, ya que permiten procesar grandes bases de datos y son capaces de manejar indisolublemente datos de altas complejidades. Los paradigmas evolutivos actuales están inspirados en la teoría de la evolución de Darwin e intentan simular en lo posible los procesos que ocurren en la naturaleza, basando la resolución de problemas computacionales en mecanismos que simulan la evolución biológica (Guillermo & Meda, 2010).

Los Algoritmos Genéticos son uno de los tipos de Algoritmos Evolutivos más utilizados, no siendo así la Programación Genética, que cuenta con menos resultados probados que estos, pero puede encontrarse entre las técnicas más prometedoras (Yager, Grichnik, & Yager, 2014). Por tanto sería de gran importancia comprobar la factibilidad del uso de la programación genética en tareas de Minería de Datos, sin embargo no existe una propuesta concreta para enfrentar, con esta técnica, la tarea de Clasificación. Las experiencias y referencias tenidas en el campo de los Algoritmos Evolutivos en la Minería de Datos no son numerosas debido entre otras razones a la novedad de su uso. Algunas referencias encontradas son:

En se describe la utilización de técnicas de Minería de Datos en sistemas de aprendizaje, presentando una herramienta visual para el descubrimiento de conocimiento en forma de reglas de predicción en la mejora de sistemas hipermedia adaptativos educativos basados en Web.

En la búsqueda de software que permitieran realizar tareas de Minería de Datos, fueron encontradas algunas herramientas entre las que se destacan DBMiner, Weka y Keel (Haupt & Haupt, 2004). Estos sistemas son de dominio

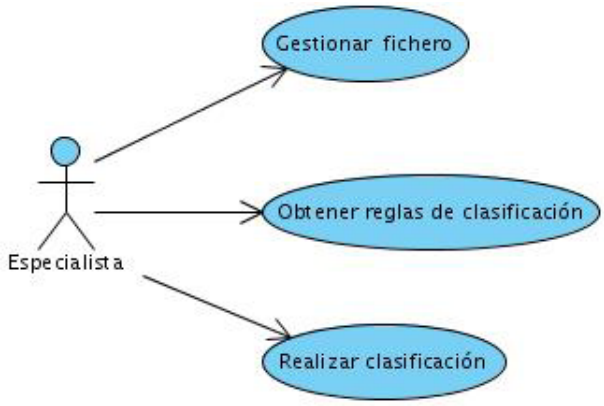
Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 10, No. 4, Mes Abril, 2017
ISSN: 2306-2495 | RNPS: 2343
Pág. 16-28
http://publicaciones.uci.cu

público un tanto populares por su entorno gráfico integrado y otra serie de características significativas. Un inconveniente que presentan estos tipos de herramientas es que son complejas de manejar para una persona no experta en minería de datos (Sonia, 2010). Además en se desarrolla un trabajo para el Descubrimiento de Reglas de Predicción mediante una herramienta gráfica denominada EPRules.

Materiales y métodos

El sistema para generación de reglas de clasificación utilizando algoritmos de programación genética en su versión 1.0, está orientado a la predicción del conocimiento de tipo biológico, el mismo cuenta con varios procedimientos que permiten la generación de reglas de clasificación para lo cual previamente se debe haber cargado un fichero con los diferentes atributos y datos correspondientes, posteriormente se debe configurar los parámetros de ejecución para la generación de las reglas así como su estructura, posteriormente se visualizan las reglas correspondiente estableciendo niveles de competencia entre ellas, posteriormente se visualizan los resultados del algoritmo de clasificación empleado.

Para comprender la propuesta, la Figura 1 muestra un diagrama de Casos de Uso del Sistema que contiene el actor, Casos de Uso del Sistema y las relaciones existentes entre los mismos. El actor Especialista biológico, es el usuario que interactúa con el sistema para validar el algoritmo implementado y así obtener las reglas de clasificación que se forman partiendo de los datos cargados previamente, posteriormente se realiza la clasificación de dichas reglas atendiendo a la aplicación del algoritmo seleccionado obteniéndose como resultado final la visualización de las reglas obtenidas del proceso de clasificación.



Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 10, No. 4, Mes Abril, 2017
ISSN: 2306-2495 | RNPS: 2343
Pág. 16-28
http://publicaciones.uci.cu

Fig. 1 Casos de uso de la propuesta

La Arquitectura de sistema consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del sistema para un sistema de información.

En el sistema llevó a cabo la arquitectura en tres capas dicha arquitectura propone un estilo de programación en la que el objetivo primordial es la separación de la lógica de negocios de la lógica de diseño, en el sistema propuesto se delimitó la capa de presentación como pantalla de presentación de los resultados de las diferentes corridas, la capa de negocio que es la que es la encargada de procesar la información así como el desarrollo del algoritmo propuesto, la capa de acceso a los datos está representada por los diferentes ficheros de los cuales se obtienen los datos para realizar las corridas, dichos ficheros son cargados y sus datos son procesados.

La ventaja principal de este estilo arquitectónico, es que el desarrollo se puede llevar a cabo en varios niveles o capas (presentación, negocio y acceso a datos) y en caso de algún cambio, sólo se ataca al nivel requerido sin tener que revisar entre código mezclado las tres capas (Guerrero, Londoño, Suárez, & Gutiérrez, 2014).

En la Figura 2, está representada la interfaz principal donde es posible realizar las principales funcionalidades definidas.

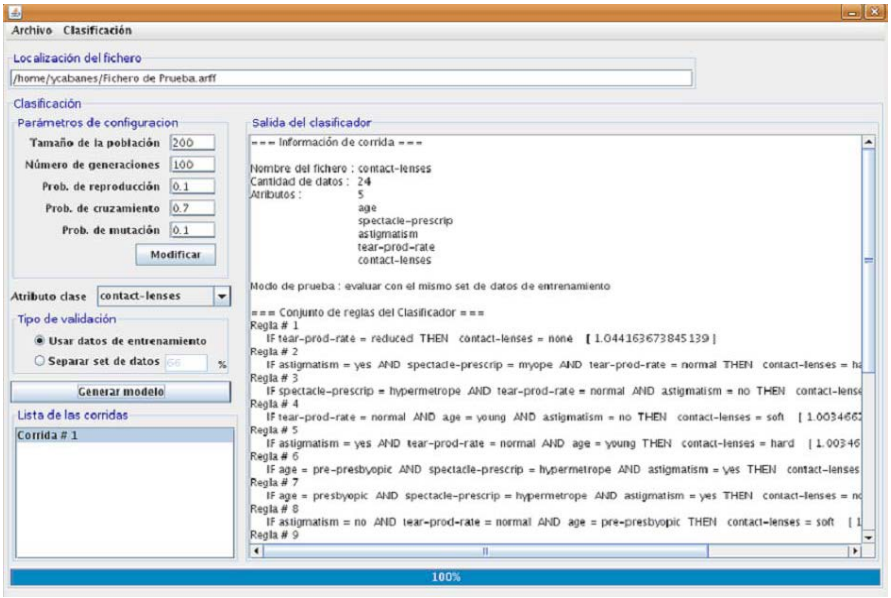


Fig. 2. Interfaz principal del sistema

La capa de negocio está representada por la Clase controladora: GPController.java Capa de acceso a datos: Es donde residen los datos y es la encargada de acceder a los datos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio. Aunque lo más usual es que haya multitud de ordenadores donde se pueda procesar cada capa individualmente, estas capas pueden residir en un único ordenador (Grau & Grau, 2012).

Esta capa está representada por los ficheros manejados así como las clases que mantienen información que será persistente: ArffFile.java y GPRule.java

El Algoritmo de clasificación implementado consta de varias fases las cuales se describen a continuación:

1. Fase de inicialización

Como se puede apreciar en la figura el algoritmo comienza generando una población inicial de individuos, o sea, reglas de clasificación, de forma aleatoria. Una regla, como se ha dicho anteriormente, está formada por un antecedente y un consecuente, y está representada mediante un árbol binario donde el subárbol izquierdo representa el antecedente y el subárbol derecho el consecuente.

Este antecedente de la regla se genera mediante el método de inicialización "Grow". (Angel, 2008) donde se va formando el árbol de forma tal que se vayan creando aleatoriamente nodos internos (no terminales) y nodos hojas (terminales) con los atributos y sus valores mientras se cumpla con la profundidad máxima establecida, todo esto garantizando que dicho árbol tenga una estructura válida para el antecedente de una regla. Luego se realiza un proceso de búsqueda sobre el conjunto de datos de entrenamiento, en el cual se localiza la clase mayoritaria en dependencia de la cantidad de datos que logre emparejar dicho antecedente y se le atribuye esta clase al consecuente formándose así la regla. Este proceso proporciona que en un inicio se generen reglas que sean capaces de cubrir al menos un dato. Seguidamente esta regla pasa a ser evaluada mediante el cálculo de la función de aptitud, donde se utilizaron dos medidas de calidad. Por un lado está el soporte (S) (Martínez, 2009) cuya forma de cálculo se muestra en la ecuación 1.

$$S = \frac{A}{T} \qquad (1)$$

Como se muestra en (1) el termino A indica el número de instancias que se cumple el antecedente de la regla y T el total de instancias que se utilizó para entrenar el modelo. Por otro lado el porcentaje de aciertos (P), reflejándose su método de cálculo en la ecuación 2.

$$P = \frac{AC}{A} \qquad (2)$$

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 10, No. 4, Mes Abril, 2017
ISSN: 2306-2495 | RNPS: 2343
Pág. 16-28
http://publicaciones.uci.cu

Como se muestra en (2) el termino AC es la cantidad de instancias que se cumple el antecedente y el consecuente de la regla. Esta medida da la probabilidad que tiene la regla de clasificar los datos correctamente. Como se decidió utilizar una función multiobjetivo, es decir, un vector de varias medidas de calidad para lograr un balance entre ellas y hacer que se converja hacia el conjunto que está formado por las mejores soluciones (en términos de todos los objetivos individuales (González, 2013), no de cada uno por separado), se forma este vector con las medidas antes mencionadas y se le calcula su norma para de esta manera darle cumplimiento al cálculo de la función de aptitud, según la ecuación 3.

$$FA = \sqrt{S^2 + P^2} \qquad (3)$$

Es válido destacar que hay otras variantes de calcular una función multiobjetivo, como es el caso de la optimización de Pareto (Salto, 2006). Por último se adiciona la regla a la población repitiéndose este proceso hasta que se cumpla con el tamaño de la población predefinido antes de la corrida del algoritmo. De esta forma queda conformada la población inicial.

2- Fase de evolución de los individuos

Después de culminada la fase uno descrita anteriormente, se pasa a evolucionar en cada generación los individuos, para de esta forma converger a una población con los mejores individuos de acuerdo al problema que se está analizando, como una característica típica de los Algoritmos Evolutivos. Para desarrollar este proceso se comienza iterando tantas veces como cantidad de generaciones se haya predefinido en los parámetros de la corrida del algoritmo(M. José & Delgado, 2012).

En cada iteración se aplica un operador genético (cruzamiento, mutación o reproducción) seleccionado aleatoriamente, en dependencia de las probabilidades que posean. Para la selección de los padres que originaran nuevos descendientes que serán incluidos en la nueva población, se utiliza el método de la ruleta. La idea que se propone en el método es dar a cada individuo una probabilidad de ser seleccionado acorde a su función de aptitud y proporcional a su calidad dentro del espacio muestral de entrenamiento que se esté analizando (cuanto mejor es el valor de la función de aptitud mayor es la probabilidad de ser seleccionado y viceversa) (Roberto & Jonatan, 2008). Después de ser aplicado el operador correspondiente se evalúan estos descendientes (de la misma forma que se describió en la fase uno), se adicionan a una nueva población y así se procede iterativamente hasta que esta última logre alcanzar el tamaño de la población predefinida.

3- Fase de evolución de las poblaciones

En el proceso iterativo desde la primera generación hasta alcanzar la última prefijada, se realiza una estrategia de corte y reemplazo de los peores individuos por otros generados, que garantiza la diversidad de la población, así como su convergencia hacia la mejor solución(Serrano, 2007).

Se comienza ordenando la población de individuos por el valor de su función de aptitud, para que se logre garantizar que los mejores individuos sean los primeros en competir por los datos a capturar y con respecto a la nueva población generada que los individuos más aptos sean los escogidos para reemplazar a los peores de la actual. Seguidamente se realiza el proceso de Competición de Tokens a la población actual ya ordenada (Ilber, 2008). En este proceso se analizan cada uno de los individuos para ver a cuántos datos puede capturar y luego, los que logran emparejar con dicho individuo, son marcados para que otro individuo menos apto que él no pueda apoderarse de este recurso. Luego se le actualiza a cada individuo el valor de la función de aptitud según la ecuación (4).

$$FM = FO \times \left(\frac{C}{M}\right) \qquad (4)$$

Donde FM es la función de adaptación modificada, FO es el valor de la función de aptitud obtenida por el individuo en el proceso de adaptación, C es el número de ejemplos que el individuo ha conseguido capturar y M es el número máximo de ejemplos que el individuo puede capturar (Mar, Zulueta, Cruz, & Leyva, 2015). Finalmente todos los individuos cuyo valor de aptitud es cero son eliminados de la población, ya que esto indica que no pudieron capturar ningún dato. Por tanto, estos k individuos son redundantes y son reemplazados por los k primeros individuos de la nueva población generada, lo que puede aportar un mayor grado de diversidad a la población y además proporcionar cambios adicionales para la generación de buenos individuos.

4- Fase de culminación

Como conclusión se vuelve a aplicar el proceso de Competición de Tokens a la última población que se obtiene, para de esta forma culminar el algoritmo con un conjunto de reglas libres de redundancias, y con alto nivel de precisión y que puede tener un número menor de reglas que el tamaño de la población predefinida (D. José, 2010). Para garantizar el correcto funcionamiento del sistema propuesto, se definen los siguientes requisitos no funcionales:

Software: Se debe disponer en las computadoras de Sistema Operativo Linux, Windows 95 o superior para la instalación de la aplicación, garantizando una interoperabilidad de forma similar en diferentes sistemas operativos o plataformas. Se necesita tener instalada la Máquina Virtual de Java nombrada Java Runtime Enviroment (JRE) versión 1.5 o superior.

Hardware: Para el desarrollo y utilización del proyecto se requieren computadoras con Procesador Pentium 3 o superior, 256 MB de RAM o más y 50 MB de capacidad de disco duro.

Usabilidad: La herramienta debe poder ser utilizada por cualquier tipo de persona que posea conocimientos básicos en el manejo de la computadora, solo se necesita que posea conocimientos de reglas de clasificación de manera que pueda entender los resultados mostrados por la misma(Garay, 2015).

Resultados y discusión

Se ha desarrollado una herramienta específica con el objetivo de facilitar el proceso de descubrimiento de reglas de clasificación y darle una aplicación al algoritmo que se propone en este trabajo. La herramienta implementada presenta una entrada de datos a procesar mediante ficheros con un formato específico (.arff). La herramienta presenta además una amigable e intuitiva interfaz visual, lo que permite que el usuario cuente con diferentes facilidades en su trabajo con la misma. A continuación se detallan estas características a las que se hace alusión.

Primeramente se puede decir que el modelo generado, o sea el conjunto de reglas de clasificación que se obtienen, es mostrado de una forma clara y precisa, acompañado de diferentes valores estadísticos que se calculan asociados al proceso de validación del modelo obtenido, como son: la cantidad de instancias correctamente clasificadas, las incorrectamente clasificadas así como las no clasificadas. Además se muestra la matriz de confusión, que es otra forma de evaluación de un modelo de clasificación (Romero, 2009), (Mar, Argota, & Santana, 2016).

Por otra parte la herramienta brinda la posibilidad de realizar varias corridas, es decir, almacena los diferentes modelos que puedan ser generados, para que el usuario puede saber en todo momento los resultados de las corridas que haya realizado y da la posibilidad de salvarlos en ficheros.

Las reglas de clasificación están compuestas por un antecedente y un consecuente donde este último posee una sola condición. Una vez que se ha generado un modelo, se tiene la posibilidad de dada las características de una nueva instancia decir a que clase pertenece, o sea, clasificarlo de acuerdo al modelo generado. Para esto primeramente se muestran, de manera dinámica, los atributos con sus respectivos valores para que estos sean seleccionados de acuerdo a las características que posea la instancia a clasificar, todo esto evita que se haga un proceso de validación de entrada de datos. Una vez obtenidos los valores de los atributos de la nueva instancia que se va a clasificar, se compara cada regla del modelo obtenido con estos nuevos valores de la instancia para ver cual empareja o está contenida y finalmente si se encuentra una o varias reglas que cumplan la condición anterior y presenten un único valor de clase, este se le asigna a la nueva instancia, de lo contrario se emite que no se puede clasificar dicha instancia.

Análisis comparativo de resultados obtenidos

Se han realizado diferentes pruebas orientadas a comparar y validar los resultados que produce el algoritmo que se propone en la tarea de descubrimiento de conocimiento, Clasificación. Para esto se determinó comprobar los resultados obtenidos, realizando dos tipos de corridas diferentes en relación con la cantidad de datos usados para entrenar y probar los modelos generados, por cada uno de los ficheros de muestra seleccionados de las Bases de Datos (Contact-Lenses, Wisconsin, Tic-Tac-Toe, Car) del UC Irvine Machine Learning Repository (Repositorio., 2013) cuyas características principales quedan resumidas en la Tabla 1. También se realizó un análisis comparativo entre el algoritmo propuesto y otros implementados en una de las herramientas estudiadas que generan modelos formados por reglas de producción y basan su mecanismo de funcionamiento mediante enfoques evolutivos.

Tabla 1: Características de los ficheros utilizados.

Nombre Fichero	No. de Atributos	No. De Clases	No. de Ejemplos
Contact-Lenses	5	3	24
Wisconsin	10	2	683
Tic-Tac-Toe	10	2	958
Car	7	4	1728

Para estos análisis se decidió tomar un tamaño de muestra de población inicial de 100, con un número de generaciones de 200 y las probabilidades de los operadores genéticos reproducción, cruzamiento y mutación fueron de 0.1, 0.7 y 0.1 respectivamente, quedando reflejados estos parámetros en la figura. 3

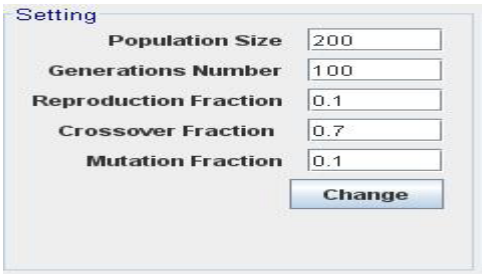


Fig. 3 Parámetros para la corrida del algoritmo.

Los tipos de corridas utilizados son los siguientes:

1. Se utiliza el mismo conjunto de datos de entrenamiento y prueba.
2. Se particiona el conjunto de datos en dos, un 50% para el entrenamiento y el resto para prueba.

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 10, No. 4, Mes Abril, 2017
ISSN: 2306-2495 | RNPS: 2343
Pág. 16-28
http://publicaciones.uci.cu

Al realizar las diferentes corridas para cada fichero analizado se obtuvieron los siguientes resultados que se expresan en la Tabla 2.

Tabla 2: Resultados de las corridas con los ficheros seleccionados.

Nombre del Fichero	Tipo de Corridas	Total Reglas	(%) Ejemplos Correctamente Clasificados	Tiempo (s)
Contact-Lenses	1	17	95.8	0.71
	2	10	33.3	0.46
Wisconsin	1	91	78.6	7.24
	2	56	78.1	4.09
Tic-Tac-Toe	1	200	64.9	11.39
	2	186	61.8	6.5
Car	1	55	70.0	16.8
	2	74	64.7	10.4

Como se puede ver en la tabla de resultados anterior, los por cientos de ejemplos correctamente clasificados, analizado para cada fichero, se comportan de forma similar en los dos tipos de corridas que se efectuaron, solo disminuye en un rango de 0.5% a 5.3%, lo que demuestra que la disposición de los datos de entrenamiento y prueba de forma variable, no afecta en grandes por cientos la precisión de clasificación.

Por otra parte se determinó realizar una comparación con dos algoritmos de clasificación que generan modelos basados en reglas de producción de la herramienta KEEL (LogenPro y PGIRLA). El primero de estos algoritmos está implementado mediante programación genética y el segundo está basado en algoritmos genéticos, por lo que ambos presentan características evolutivas propias muy parecidas al que se trata en este trabajo, lo que facilita una mejor comparación. Al realizar estas pruebas se obtuvieron los resultados que se muestran en la Tabla 3.

Tabla 3: Resultados de las corridas con los algoritmos y ficheros seleccionados.

Tabla 3: Resultados de las corridas con los algoritmos y ficheros seleccionados.

Algoritmos	Contact-Lenses	Wisconsin	Tic-Tac-Toe	Car
Algoritmo propuesto	95.8	78.6	64.9	70
LogenPro	83.3	62.8	69.2	69.2

PGIRLA	58.3	78.4	56.2	56.2

Como se puede apreciar en las dos tablas anteriores, los resultados obtenidos con el algoritmo que se propone en cuanto a la medida de precisión de clasificación que se presenta, comparándolos con los que arrojan LogenPro y PGIRLA son muy similares y en tres casos arroja los mejores resultados, lo que da una medida de validez del algoritmo que se propone.

Conclusiones

Para garantizar la extracción de conocimiento como parte del proceso de minería de datos en análisis biológicos, se requiere de novedosos algoritmos que implementen alta eficiencia en sus resultados.

Con la introducción de Algoritmos Evolutivos, enfocados a la Programación Genética es posible potenciar resoluciones para la Minería de Datos que contribuya a la extracción de conocimientos.

Con la introducción en la práctica social del algoritmo para la generación de reglas de clasificación basado en programación genética. Se pudo realizar un análisis comparativo de los resultados obtenidos con otros algoritmos existentes demostrándose la efectividad de la propuesta.

Referencias

Angel, G. (2008). Computación Evolutiva (CE), Programación Genética, Evolución Gramatical, Programación por Expresión Genética. *Escuela de Ingeniería de Sistemas y Computación, Universidad del Valle*, 20-45.

Garay, M. (2015). Interfaces Inteligentes en el aprendizaje de la Modelación. *Ingeniería Industrial, Vol. XXXVI*(No. 2), 187-201.

González, J. (2013). Propuesta de algoritmo de clasificación genética. *RCI, Vol. 4* (No.2), 37-42.

Grau, I., & Grau, R. (2012). Aplicación de sistemas neuroborrosos a problemas de resistencia antiviral del VIH. *RCCI, Vol.6*(No2).

Guerrero, C., Londoño, J., Suárez, J., & Gutiérrez, L. (2014). Estudio comparativo de marcos de trabajo para el desarrollo software orientado a aspectos. *Inf. tecnol, vol.25* (no.2).

Guillermo, M., & Meda, M. (2010). Integración de Minería de datos y Sistemas Multiagente: un campo de investigación y desarrollo. *Ciencias de la Información, Vol41*(No3), 53-56.

Haupt, R., & Haupt, S. (2004). Practical Genetic Algorithms , John Wiley & Song, 2nd ed, Canada Month J. A. *Electric power system applications of optimization, Editorial CRC Press,Estados Unidos.*

Ilber, A. (2008). DESIGN OF A ADAPTIVE CONTROLLER BASED IN GENETIC ALGORITHMS. *Revista Colombiana de Tecnologías de Avanzada, VOl2*(No6).

José, D. (2010). Un algoritmo genético híbrido y un enfriamiento simulado para solucionar el problema de programación de pedidos job shop. *Revista EIA, Vol13*(39-51).

José, M., & Delgado, R. (2012). Algoritmo genético aplicado a la programación en talleres de maquinado Ingeniería Mecánica. *Vol15*(No3).

Mar, O., Argota, L., & Santana, I. (2016). Módulo para la evaluación de competencias a través de un Sistema de Laboratorios a Distancias. *RCCI, Vol.10*(No.2), 132-147.

Mar, O., Zulueta, Y., Cruz, M., & Leyva, M. (2015). Motor de inferencia decisional en sistema informático para la evaluación del desempeño. *RCCI, Vol.9*(No.4), 16-29.

Martínez, C. (2009). Selección de medidas de evaluación de reglas obtenidas mediante programación genética basada en gramática. *Universidad de Córdoba.* Retrieved 20 de septiembre 2015, from www.lsi.us.es/redmidas/Capitulos/LMD23.pdf

Pistohl, T., Joshi, D., Ganesh, G., Jackson, A., & Nazarpour, K. (2015). Artificial Proprioceptive Feedback for Myoelectric Control. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on, 23*(3), 498-507. doi: 10.1109/TNSRE.2014.2355856

Repositorio. (2013). Bases de Datos UCI. Retrieved 20 de septiembre del 2015, from http://archive.ics.uci.edu/ml/

Roberto, P., & Jonatan, P. (2008). Un algoritmo genético paralelo que combina los modelos de grillas e islas para encontrar soluciones óptimas cercanas al problema del agente viajero. *Revista en avaces en Sistemas e Informática, Vol5*(No3).

Romero, C. (2009). Aplicación de algoritmos evolutivos como técnica de minería de datos para La mejora de cursos Hipermedia adaptativos basados en Web *RIED. Revista Iberoamericana de Educación a Distancia, Vol.6*(No.2).

Salto, A. M. (2006). Analysis of Distributed Genetic Algorithms for Solving Cutting Problems. (), Volume 13/issue number 5, pág . *ITOR, Vol.13*(No.5), p. 403-423.

Serrano, G. (2007). Herramienta para la generación de reglas de asociación basada en un algoritmo de Programación Genética. *Tesis, Ciudad de la Habana*, 24-60.

Sonia, O. (2010). Aprendizaje de Reglas Encadenas para la Creación de Grafos Conceptuales. *Polibits, No4.*

Yager, R. R., Grichnik, A. J., & Yager, R. L. (2014). A Soft Computing Approach to Controlling Emissions Under Imperfect Sensors. *Systems, Man, and Cybernetics: Systems, IEEE Transactions on, 44*(6), 687-691. doi: 10.1109/TSMC.2013.2268735