UCLV
Universidad Central
"Marta Abreu" de Las Villas

MFC
Facultad de Matemática
Física y Computación

Laboratorio de Inteligencia

TRABAJO DE DIPLOMA

Título: Biblioteca para el preprocesamiento de datos usando conjuntos aproximados

Autora: Beatriz Bello García

Tutores: Dr.C Rafael Bello Pérez

Dr.C Carlos Morell Pérez

Consultante: Dra.C María M García Lorenzo

Santa Clara, junio 2019

UCLV
Universidad Central
"Marta Abreu" de Las Villas

MFC
Facultad de Matemática
Física y Computación

Artificial Intelligence Laboratory

DIPLOMA THESIS

Title: Library for the preprocessing of data using approximate sets

Author: Beatriz Bello García

Thesis Director: Dr.C Rafael Bello Pérez

Dr.C Carlos Morell Pérez

Other advicer: Dra.C. María M García Lorenzo

Santa Clara, June, 2019

PENSAMIENTO

Todos tenemos sueños. Pero para convertir los sueños en realidad, se necesita una gran cantidad de determinación, dedicación, autodisciplina y esfuerzo.

Jesse Owens

DEDICATORIA

A mis padres

AGRADECIMIENTOS

A mi mamita preciosa que es mi guía y mi fuerza. Mi consejera y defensora, mi amiga, mi saco de lágrimas y mi razón de ser. Porque llora cuando yo lloro, porque sufre cuando yo sufro, porque ríe porque yo río, porque es indispensable para mí gracias. Gracias por dedicarte a cuidarnos, protegernos, velar por que nada falte, ser el faro y motor impulsor de toda la familia. Gracias por ser mi mama y la persona más especial en mi vida.

A mi gordo querido, mi papa, por su sacrificio y esfuerzo, por sus consejos y sus ideas. Por enseñarme a apreciar la vida, por hacer de mí una persona responsable e independiente, por enseñarme a valorar las oportunidades y a anteponerme a los problemas. Por ser mi ejemplo a seguir y demostrarme que cuando se quiere se puede. Porque me impulsa a lograr mis sueños y a no rendirme. Por querer y proteger a toda la familia.

Gracias a los dos por tanto sacrificio, dedicación y sobre todo por dejarme ser parte de sus vidas.

A mi hermana por compartir conmigo sus locuras y disparates, porque aunque a veces nos llevemos como el perro y el gato ha estado a mi lado en los momentos buenos y malos, en la que confió.

A mi abuela Adelfa por sus grandes abrazos, su apoyo, su comprensión y cuidado. Porque me contagia su optimismo y alegría.

A mi tía Dunia por ser como una segunda mama para mí y complacerme en mis caprichos y deseos, por darme ese hermano varón, para que me proteja y me acompañe Jesús Rafael.

A mi abuelo Chury por sus enseñanzas y sus historias, por cada uno de los momentos que pase a su lado. Porque aunque hoy no esté presente sé que estaría muy orgulloso y feliz.

A mi prima Yaima por ser como una hermana para mí, en la que puedo confiar y apoyarme siempre.

A mi abuela Tete por sus cariños y cuidados, por su atención y dulzura. A mi tío Julio por lograr que me guste la historia, por ser tan atento y dedicado.

A Ale por formar parte de mi vida, por ser tan protector y detallista. Por toda su ayuda y comprensión. Por estar a mi lado a pesar de mis resabios, porque me siento segura a su lado. Gracias por compartir conmigo tus deseos y preocupaciones. A Esther y a tus padres por su cariño.

A Pancho y Mary que son mis padrinos y mis camaroncitos duros. Gracias por su protección y cuidados.

A todos los amigos que conoci estos 5 años de carrera, en especial a Robe, Nede, Abel, Tato Giselle, Rafa, Ruddy, Raikol, Pancho, Felix, Maria Karla y Rocío por todas las noches de estudios y los momentos que compartimos.

A las amistades que siguen presentes a pesar de los años y con las que sé que puedo seguir contando Claudia, Anny y Eliani.

A mis profesores, en especial a Leticia por asesorarme y enseñarme los tres años de alumna ayudante.

A Morell y a mi papa por querer ser mis tutores. A Ricardo por ser mi asesor y consultante en esta última etapa de la tesis, ayudándome siempre en lo que necesite.

A la virgen de la Caridad del Cobre y los ángeles por su protección.

A todos los que han formado parte del desarrollo de esta tesis de una forma u otra.

GRACIAS!!!

RESUMEN

La comunidad científica reconoce el papel de la Teoría de los Conjuntos Aproximados (*Rough Set Theory*, RST) para el análisis inteligente de los datos. En este trabajo, se describe la implementación en Python de una biblioteca para el preprocesamiento de datos, etapa previa determinante para el descubrimiento de conocimiento, de métodos basados en los conjuntos aproximados. En particular, son implementadas las definiciones y medidas principales basadas en RST, así como métodos de selección de rasgos y ejemplos. Se verifica la eficacia de los métodos a partir de las pruebas realizadas desde bases de datos del UCI repositorio.

Además, se tratan los conceptos básicos de RST para grandes volúmenes de datos, en particular, lo referido a clase de equivalencia. Se implementa la vectorización de código utilizando las facilidades de Numpy de Python. Por último, se implementaron algunos conceptos básicos en entorno Apache Spark. Los resultados demuestran la validez de las implementaciones y la necesidad de optimizar código.

ABSTRACT

The scientific community recognizes the role of the Rough Set Theory (RST) for the intelligent data analysis. In this paper, we describe the Python implementation of a library for data preprocessing, a decisive preliminary stage for data mining. In particular, the definitions and main measures based on RST are implemented, as well as the methods of selection of features and examples. The accuracy of the methods is verified from the tests performed using databases from the repository UCI.

In addition, the basic concepts of RST for large volumes of data are discussed, in particular, referring to compute the equivalence classes. Code vectorization is implemented using the Numpy facilities of Python. Finally, some basic concepts were implemented in the Apache Spark environment. The results demonstrate the validity of the implementations and the need to optimize code.

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTAS DE TABLAS

INTRODUCCIÓN

En la actualidad el desarrollo de las nuevas tecnologías posibilita la acumulación de grandes volúmenes de datos. El verdadero valor de los datos radica en la posibilidad de extraer de ellos información útil para la toma de decisiones o la exploración y comprensión de los fenómenos que le dieron lugar (Ruiz 2006). El análisis de datos es importante en ramas como: bioinformática, medicina, economía y finanzas, industria, medio ambiente, entre otras. Sin embargo, la disponibilidad de los volúmenes de datos e información genera la necesidad de desarrollar procesos de aprendizaje para la extracción y descubrimiento de conocimiento.

La Inteligencia Artificial constituye uno de los subcampos de la Ciencia de la Computación que se encarga de dar solución a problemas que no tienen un algoritmo específico para resolverlos, y precisamente se trata de aquellos problemas que involucran el razonamiento, la experiencia acumulada de expertos humanos, el aprendizaje, entre otras. La Inteligencia Artificial es el estudio de las facultades mentales a través del uso de los modelos computacionales (Russell and Norvig, 2016).

Una de las áreas de la Inteligencia Artificial que se ocupa de desarrollar técnicas capaces de aprender, es decir, extraer de forma automatizada conocimiento subyacente en la información es el Aprendizaje Automatizado (del inglés, "*Machine Learning*"), la cual constituye, junto con la estadística, el corazón del análisis inteligente de los datos. Tom Mitchell define el aprendizaje automatizado como: Se dice que un programa de computadora aprende de la experiencia E con respecto a alguna clase de tareas T y la medida de rendimiento P, si su desempeño en las tareas en T, medido por P, mejora con la experiencia E (Mitchell 1997).

Existen diferentes tipos de aprendizaje; clasificándose en supervisado, no supervisado, y semisupervisado dependiendo de la muestra de aprendizaje que se tenga. A su vez, se desarrollan distintos tipos de métodos que facilitan la minería de los datos o el descubrimiento de conocimiento y que enmarcan diversas funciones como son: la clasificación, la predicción numérica, la asociación o el agrupamiento. Para el desarrollo de estos procesos es imprescindible después de identificado el problema preprocesar la información o la muestra de datos de donde se quiere aprender.

El preprocesamiento de datos, engloba a todas las técnicas de análisis de datos que permite mejorar la calidad de los datos para que las técnicas de extracción de conocimiento o minería de datos puedan obtener mayor o mejor información. La preparación de los datos puede generar un conjunto más pequeño de datos que el original con una mayor calidad a partir de técnicas como selección de datos relevantes, eliminación de anomalías, eliminación de datos duplicados, etc. (Hernández and Rodriguez, 2013).

La calidad del conocimiento extraído depende en gran medida de la calidad de los datos. Desgraciadamente, estos datos se ven afectados por factores negativos como: ruido, valores perdidos, inconsistencias, datos superfluos y/o un tamaño demasiado grande en cualquier dimensión (número de atributos e instancias). Está demostrado que una baja calidad de los datos conduce en la mayoría de los casos a una baja calidad del conocimiento extraído (García 2015). De modo que es importante desarrollar métodos que faciliten el procesamiento de datos y utilizar adecuadamente los que ya existen.

Una de las teorías empleadas para el análisis inteligente de los datos es la Teoría de los Conjuntos Aproximados (*Rough Set Theory*, RST) (Pawlak 1982; Komorowski and Pawlak 1999; Bello and Nowe et al. 2006). Esta teoría se considera como una de las cinco áreas claves y no tradicionales de la Inteligencia Artificial y de la Teoría de la Información Incompleta, pues constituye una herramienta muy útil para el manejo de la información no completa o imprecisa (Chin, Liang et al. 2003; Peters 2005). La RST se ha usado para la generación de reglas, la selección de atributos (Dunstsh and Gunter 2000; Zhong, Dong et al. 2001), entre otras aplicaciones (Caballero, Bello et al. 2007)

RST se define a partir de dos componentes básicas: un Sistema de Información o un Sistema de Decisión y una relación de inseparabilidad (Skowron 1999). La primera de estas componentes se corresponde con los conjuntos de aprendizajes utilizados por los métodos de aprendizaje automático; mientras que la segunda componente permite trabajar con el enfoque de la computación granular.

Recientemente ha emergido el termino de Big Data o datos masivos el cual se refiere a una colección de datos grande, complejos, muy difícil de procesar a través de herramientas de gestión y procesamiento de datos tradicionales, cuyo volumen, diversidad y complejidad requieren nueva arquitectura, técnicas, algoritmos y análisis para gestionar y extraer valor y

conocimiento oculto en ellos (Fernández *et al.*, 2014); su objetivo es filtrar el ruido y mantener los datos valiosos, que pueda utilizarse para la toma de decisiones inteligentes.

Big Data es un área emergente y en expansión. Las posibilidades de desarrollo de algoritmos para nuevos datos y, aplicaciones reales es un nicho de investigación y desarrollo en los próximos años. No hay dudas de que el progreso y la innovación ya no se ven obstaculizados por la capacidad de recopilar datos, sino por la capacidad de gestionar, analizar, sintetizar, visualizar, y descubrir el conocimiento de los datos recopilados de manera oportuna y en una forma escalable (Russom and Org, 2011).

Con el desarrollo del Aprendizaje Automatizado se han desarrollado múltiples herramientas, plataformas o bibliotecas que incorporan distintos métodos atendiendo a la función a minar. Entre ellas están: WEKA, MEKA, KEEL, MULAN (Witten, Frank, Hall and Pal, 2017), etc. Todas ellas incluyen distintos métodos para el preprocesamiento de datos tales como filtros, discretizadores, selectores, etc.

Sin embargo, en la mayoría de las herramientas para el descubrimiento de conocimiento no se incluyen métodos basados en conjuntos aproximados. Por tal razón, el **problema de investigación** viene dado por las insuficiencias de herramientas computacionales para el pre-procesamiento de datos usando conjuntos aproximados, incluido el entorno de Big Data.

Objetivo general

Implementar técnicas para el preprocesamiento de datos usando métodos basados en conjuntos aproximados en el entorno Python.

Objetivos específicos

- ✓ Crear una biblioteca en el lenguaje Python para el preprocesamiento de datos utilizando métodos para la selección de rasgos y edición de conjuntos de datos basados en conjuntos aproximados.
- ✓ Introducir las técnicas de vectorización en la implementación de las técnicas para el pre-procesamiento de datos basadas en conjuntos aproximados para incrementar la eficiencia de las mismas.
- ✓ Diseñar como implementar los principales conceptos de la Teoría de los conjuntos aproximados en el entorno Spark.

Hipótesis

La implementación de herramientas para el procesamiento de datos usando la teoría de los conjuntos aproximados y su implementación en Python y Spark facilita el desarrollo de aplicaciones de aprendizaje automatizado clásico y en el entorno Big Data.

La tesis se estructura en tres capítulos:

El capítulo 1 corresponde al marco teórico referencial en donde se profundiza en la RST para el análisis inteligente de datos y su utilidad en el preprocesamiento de estos. El capítulo 2 corresponde a la implementación de la biblioteca para preprocesamiento de datos en RST detallando cada uno de los métodos utilizados. Por último, en el capítulo 3 se realiza un análisis sobre el tratamiento de los conceptos de la RST para grandes volúmenes de datos.

Capítulo 1

Análisis inteligente de datos usando la teoría de los conjuntos aproximados

1. ANÁLISIS INTELIGENTE DE DATOS USANDO LA TEORÍA DE LOS CONJUNTOS APROXIMADOS

Inmensas cantidades de información nos rodean en la actualidad. Tecnologías como Internet generan datos a un ritmo exponencial gracias al abaratamiento y gran desarrollo del almacenamiento y los recursos de red. La información en sí misma tiene pocas ventajas, su sistematización, incorporación y utilización son los elementos que aportan su valor añadido: el conocimiento. Es necesario crear sistemas que generen conocimiento, para asegurar el uso productivo de la información y guiar una toma de decisiones óptima, contribuyendo de esta forma a la Gestión del Conocimiento (Bueno 2001; Canals 2003; Frappaolo 2006).

En los últimos años, el crecimiento masivo en la escala de los datos está siendo un factor clave en el actual escenario de procesamiento de datos. La eficacia de los algoritmos de extracción de conocimiento depende en gran medida de la calidad de los datos, la cual puede ser garantizada por los algoritmos de preprocesamiento (Memon *et al.*, 2017).

Este capítulo constituye el marco referencial de la tesis donde se aborda lo relativo al preprocesamiento de datos y se profundiza en la Teoría de los Conjuntos Aproximados para el análisis inteligente de los datos.

1.1 Inteligencia Artificial y el análisis inteligente de los datos

La Inteligencia Artificial (IA) es la rama de las Ciencias de la Computación que intenta reproducir los procesos de la inteligencia humana mediante el uso de las computadoras (Russell and Norvig, 1995). El primer intento de definir la Inteligencia Artificial lo hizo el matemático Alan Turing, a través del Test de Turing al expresar que “si durante el intercambio entre una computadora y el usuario este último cree que está intercambiando con otro humano, entonces se dice que el programa es inteligente”. Un primer intento de conseguir que las maquinas razonarán se llevó a la práctica mediante los llamados sistemas expertos, los que tratan de llegar a conclusiones lógicas a partir de hechos o premisas introducidas a priori en el sistema (García, 2012). Actualmente se utilizan técnicas más versátiles como las redes probabilísticas, las redes neuronales artificiales y los sistemas basados en casos que permiten hacer predicciones y arribar a conclusiones incluso cuando hay cierto nivel de incertidumbre en las premisas.

El aprendizaje automático es también una condición necesaria para que un ente artificial sea considerado inteligente**.** Una de las críticas que se oyen más a menudo respecto a la IA, es que las máquinas no se pueden considerar inteligentes hasta que no sean capaces de aprender a hacer cosas nuevas y a adaptarse a las nuevas situaciones, en lugar de limitarse a hacer aquellas actividades para las que fueron programadas. En vez de preguntarse si una computadora es capaz de "aprender", resulta mucho más clarificador intentar describir a qué actividades se refiere exactamente cuándo se dice "aprender " y cuáles mecanismos se pueden utilizar para llevar a cabo dichas actividades. Herbert Simon define aprendizaje como cualquier cambio en un sistema que le permite a este resolver mejor una tarea la segunda vez u otra tarea similar de manera más eficiente y eficaz cada vez. (Simon 1983). Si una máquina no es capaz de aprender cosas nuevas, difícilmente será capaz de adaptarse al medio.

1.1.1 Aprendizaje Automatizado

Al subcampo de la Inteligencia Atificial que estudia los métodos de solución de problemas de aprendizaje por las computadoras se le denomina Aprendizaje Automatizado (*Machine Learning*, ML).

El aprendizaje puede tener diferentes fuentes de conocimiento. Una puede ser el propio humano, que actúe como tutor para que la computadora aprenda (por ejemplo, un concepto). Otra puede ser un conjunto de ejemplos de problemas resueltos del dominio de aplicación. El resultado del aprendizaje puede ser conocimiento explícito o implícito. Se dice explícito cuando producto del proceso de aprendizaje se obtiene conocimiento en alguna forma, por ejemplo reglas u operadores; ejemplo de este tipo de aprendizaje es el algoritmo ID3 (Mitchell 1997) y sus extensiones C4.5 (Quinlan 1993) y J48 (Bhargava *et al.*, 2013). Se denomina implícito cuando no se obtiene conocimiento explícito desde el conjunto de ejemplos pero este sirve para que la computadora pueda resolver nuevos problemas o alcanzar mejores soluciones; también denominado Aprendizaje Perezoso (*Lazy learning)* (Aha, 1997), el k-NN (Cover and Hart 1967) es un típico método de aprendizaje perezoso.

El problema del aprendizaje se puede abordar desde diferentes disciplinas. Entre ellas la Estadística inferencial (distinguirla de la Estadística descriptiva en la cual se hallan magnitudes como promedios, varianza, moda, etc.), las Redes Neuronales Artificiales

(Witten and Frank 2005a), las Redes Bayesianas (Sucar, 2006), la Programación Lógica Inductiva, etc.

Estas disciplinas han desarrollado diversas técnicas, entre ellas inducción de árboles de decisión, inducción de reglas, aprendizaje basado en ejemplos, clasificadores bayesianos, Reglas neuronales artificiales (Borgelt and Kruse 2003) y Algoritmos Genéticos.

En (Verdenius and van Someren, 1997) se usa el término Técnicas de Aprendizaje Inductivo para aquellas técnicas que extraen modelos como generalizaciones a partir de los datos, las cuales han recibido un fuerte reforzamiento a partir del desarrollo de la Minería de Datos. Estas técnicas se clasifican en las clases siguientes: Inducción de árboles de decisión reglas (caso ID3 y C4.5 (J48)) donde las generalizaciones se presentan como árboles de decisión o reglas, Algoritmos de aprendizaje de las redes neuronales artificiales (caso Multilayer Perceptron y Red de Kohonen) donde las generalizaciones se presentan en forma de redes de unidades de procesamiento interconectadas por enlaces pesados, etc.

Los métodos de Aprendizaje Automatizado se pueden clasificar en supervisados y no supervisados. En el caso supervisado se conoce la clase a la que los ejemplos pertenecen y el sistema aprende a cómo clasificar nuevos casos. En el no supervisado se tiene que descubrir los conceptos o clases a los cuales pertenecen los ejemplos.

Se puede afirmar que el Aprendizaje Automatizado es el área de la Inteligencia Artificial que se ocupa de desarrollar técnicas capaces de aprender, es decir, extraer de forma automatizada conocimiento subyacente en la información. Constituye, junto con la estadística, el corazón del análisis inteligente de los datos (Ruiz 2006).

El proceso de extraer conocimiento a partir de bases de datos se conoce como KDD (*Knowledge Discovery in Databases*) (Fayyad, Piatetsky-Shapiro and Smyth, 1996). Este proceso comprende diversas etapas, que van desde la obtención de los datos hasta la aplicación del conocimiento adquirido en la toma de decisiones. Entre esas etapas, se encuentra la que puede considerarse como el núcleo del proceso KDD y que consiste en la extracción del conocimiento a partir de los datos. Esta fase es crucial para la obtención de resultados apropiados, y depende del algoritmo de Aprendizaje Automatizado que se aplique. Pero se ve influida en gran medida por la calidad de los datos que llegan desde la fase previa (Ruiz 2006).

En el mundo real, los datos frecuentemente no están limpios, contienen inconsistencias y suelen mostrar ruido, conteniendo errores, valores atípicos, ausencia de información, etc. Sin un preprocesamiento de datos puede disminuir la calidad de la minería de datos. El preprocesamiento de los datos es un paso muy importante en el proceso de minería de datos o extracción de datos y en el Aprendizaje Automatizado en general.

1.2 Preprocesamiento de datos

Grandes cantidades de datos están a nuestro alrededor, datos sin procesar que son principalmente intratables para aplicaciones humanas o manuales y que con un tratamiento adecuado pueden llegar a convertirse en conocimiento y fuente de retroalimentación para futuros procesos de extracción de datos. Todo esto provoca que el análisis y procesamiento de estos sea ahora una necesidad. El preprocesamiento de datos tiene como objetivo principal la obtención de un conjunto de datos final que sea de calidad y útil para la fase de extracción de conocimiento (García 2016).

El preprocesamiento de datos incluye la preparación de datos, compuesto por la integración, limpieza, normalización y transformación de datos; y tareas de reducción de datos; como selección de características, selección de ejemplos, discretización, etc. Debido a que en muchas ocasiones los datos provienen de diferentes fuentes, pueden contener valores incompletos (ausencia de información), con ruido (error aleatorio o variación en el valor de un atributo, debido normalmente a errores en la medida del mismo) e inconsistentes, y esto puede conducir a la extracción de patrones poco útiles. El resultado esperado después de un encadenamiento confiable de tareas de preprocesamiento de datos es un conjunto de datos que puede considerarse correcto y útil para otros algoritmos de minería de datos

La preparación o preprocesamiento de datos engloba a todas aquellas técnicas de análisis de datos que permiten mejorar la calidad de un conjunto de ellos, de modo que los métodos de extracción de conocimiento puedan obtener mayor y mejor información. Los procesos que se incluyen en esta fase se pueden resumir en: recopilación de datos, limpieza, transformación y reducción, y no siempre se aplican en un mismo orden (ver figura 1.1).

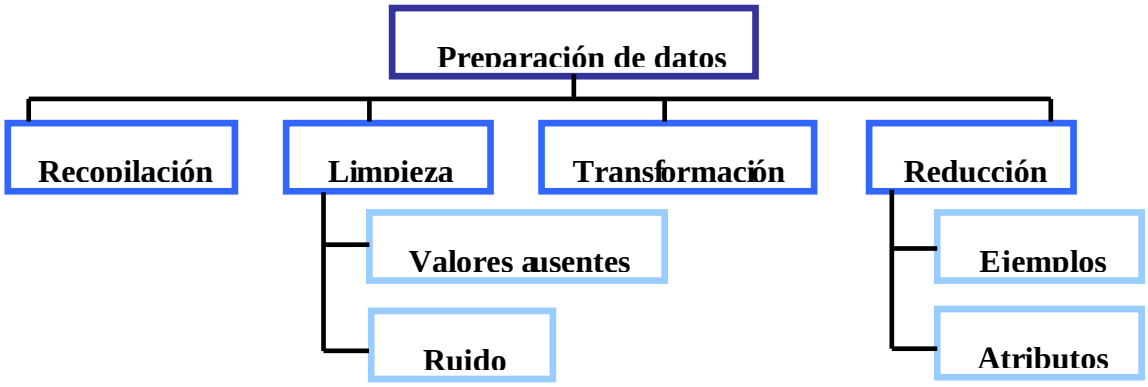

Figura 1.1 Fases de preparación o preprocesamiento de los datos

Para poder comenzar a analizar y extraer algo útil en los datos es preciso, en primer lugar, disponer de ellos. Esto en algunos casos puede parecer trivial, a partir de un simple archivo de datos, sin embargo en otros, es una tarea muy compleja dado por problemas de representación, de codificación e integración de diferentes fuentes para crear información homogénea, este es el llamado proceso de recopilación. En la fase de limpieza se resuelven conflictos entre datos, y se comprueban problemas de ruido, valores ausentes, entre otros.

En ocasiones, la forma en que viene dada la información originalmente no es la más adecuada para adquirir conocimiento a partir de ella. En esas situaciones se hace necesaria la aplicación de algún tipo de transformación para adecuar los datos al posterior proceso de aprendizaje, como por ejemplo normalización o cambio de escala, discretización, generalización o extracción de atributos. Esta última transformación está estrechamente relacionada con la selección de características, y consiste en extraer conjuntos de atributos a partir de combinaciones de los originales. Es conveniente, aplicar técnicas de reducción al conjunto de datos, orientadas fundamentalmente hacia dos objetivos: técnicas de selección de atributos (eliminar aquellos atributos que no sean relevantes para la información inherente al conjunto de datos) y técnicas de edición (reducción del número de objetos o ejemplos para mejorar el desempeño de los clasificadores con menos costo computacional). En la figura 1.1 se pueden observar los dos tipos de reducción.

La limpieza de los datos, incluye la operación que corrige los datos del conjunto de datos, filtra los datos incorrectos y reduce los detalles innecesarios de los datos. También la

detección de discrepancias y datos sucios (fragmentos de los datos originales que no tienen sentido) (Fernando, Saldaña and Flores, 2005).

Los datos se convierten o se consolidan para que el resultado del proceso de minería pueda aplicarse o puede ser más eficiente en el proceso de transformación de los datos. Las subtareas dentro de la transformación de datos son la normalización, la discretización, la generalización. La mayoría de ellos se segregarán como tareas independientes, debido al hecho de que la transformación de datos, como el caso de la limpieza de datos, se conoce como una familia general de técnicas de preprocesamiento de datos.

El proceso de discretización transforma los datos cuantitativos en datos cualitativos, es decir, los atributos numéricos se clasifican con un número finito de intervalos, obteniendo una partición superpuesta de un dominio continuo. Luego se establece una asociación entre cada intervalo con un valor numérico discreto. Una vez que se realiza la discretización, los datos se pueden tratar como datos nominales durante cualquier proceso de minería de datos (García, Luengo and Herrera 2015).

La reducción de datos comprende el conjunto de técnicas que, de una manera u otra, obtienen una representación reducida de los datos originales (Herrera and Cano, 2006). En el caso de la reducción de datos, los datos producidos generalmente mantienen la estructura e integridad esenciales de los datos originales, pero la cantidad de datos se reduce. La reducción puede darse a partir de la selección de ejemplos y o la selección de rasgos.

La selección de ejemplos consiste en una selección aleatoria de ejemplos, se conoce habitualmente como un muestreo, o la elección del mejor subconjunto posible de ejemplos a partir de los datos originales mediante el uso de algunas reglas y / o heurísticas.

.

1.3 Análisis Inteligente de datos usando Teoría de los conjuntos aproximados

La Teoría de los Conjuntos Aproximados abrió una nueva dirección en el desarrollo de teorías sobre la información incompleta y es una poderosa herramienta para el análisis de datos. Esta teoría es de gran utilidad en el preprocesamiento de los datos, tanto para encontrar conjuntos reducidos de atributos y editar conjuntos de entrenamiento para resolver problemas

de clasificación supervisada, así como para generar conocimiento a priori sobre un conjunto de datos (Skowron and Peters 2003).

Se basa en aproximar cualquier concepto, un subconjunto duro del universo, como por ejemplo, una clase en un problema de clasificación supervisada, por un par de conjuntos exactos, llamados Aproximación Inferior y Aproximación Superior del concepto. Con esta teoría es posible tratar tanto datos cuantitativos como cualitativos, y no se requiere eliminar las inconsistencias previas al análisis; respecto a la información de salida, puede usarse para determinar la relevancia de los atributos, generar las relaciones entre ellos, entre otras. Por otro lado, provee determinadas medidas para evaluar la calidad del conjunto de entrenamiento entre las cuales se encuentra la inconsistencia que describe una situación en la cual hay dos o más valores en conflicto para asignarse a una variable (Parsons 1996).

Varios autores ven esta teoría como la mejor herramienta para modelar la incertidumbre cuando esta se manifiesta en forma de inconsistencia, y como una nueva dirección en el desarrollo de teorías sobre la información incompleta (Grzymala-Busse 1994; Greco 2001).

A continuación se describen los conceptos fundamentales de la RST.

1.3.1 Principales definiciones de la Teoría de los Conjuntos Aproximados

La filosofía de los conjuntos aproximados se basa en la suposición de que con todo objeto $\boldsymbol{x}$ de un universo $\boldsymbol{U}$ está asociada una cierta cantidad de información (datos y conocimiento), expresado por medio de algunos atributos que describen el objeto (Komorowski and Pawlak 1999).

Diversos modelos computacionales operan sobre colecciones de datos. En cada caso esta colección tiene sus características, sobre todo organizativas, y recibe una denominación particular. En el caso de la Teoría de los Conjuntos Aproximados la estructura de información básica es el Sistema de Información.

Definición 1. Sistema de Información y sistema de decisión

Sea un conjunto de atributos $A= \{a_1, a_2, \ldots, a_n\}$ y un conjunto U no vacío llamado universo de ejemplos (objetos, entidades, situaciones o estados) descritos usando los atributos a_i; al par (U, A) se le denomina Sistema de información (Komorowski and Pawlak 1999). Si a

cada elemento de *U* se le agrega un nuevo atributo *d* llamado decisión, indicando la decisión tomada en ese estado o situación, entonces se obtiene un Sistema de decisión (U, A∪ {d}), donde $d \notin A$.

Definición 2. Función de información

A cada atributo a_i se le asocia un dominio v_i. Se tiene una función $f: UxA \rightarrow V$, $V = \{v_1, v_2, \ldots, v_p\}$ tal que $f(x,a_i) \in v_j$ para cada $a_i \in A$, $x \in U$, llamada función de información (Komorowski and Pawlak 1999).

El atributo de decisión *d* induce una partición del universo *U* de objetos. Sea el conjunto de enteros $\{1, \ldots, l\}$, $X_i = \{x \in U: d(x)=i\}$, entonces $\{X_1, \ldots, X_l\}$ es una colección de clases de equivalencias, llamadas clases de decisión, donde dos objetos pertenecen a la misma clase si ellos tienen el mismo valor para el atributo decisión.

Se dice que un atributo $a_i \in A$ separa o distingue un objeto *x* de otro *y*, y se escribe *Separa* (a_i, x, y), si y solo si se cumple:

$$f(x, a_i) \neq f(y, a_i) \quad (1)$$

La relación de separabilidad se basa en la comparación de los valores de un atributo, para lo cual se usa la igualdad (o desigualdad) estricta. Sin embargo, es posible usar una condición de comparación menos estricta definida de esta forma:

$$Separa\ (a_i, x, y) \Leftrightarrow f(x, a_i) - f(y, a_i) > \varepsilon \quad (2)$$

Definición 3. Relación de inseparabilidad

A cada subconjunto de atributos *B* de *A*, $B \subseteq A$, está asociada una relación binaria de inseparabilidad denotada por *R*, la cual es el conjunto de pares de objetos que son inseparables uno de otros por esa relación (Komorowski and Pawlak 1999).

$$R = \{(x, y) \in UxU: f(x, a_i) = f(y, a_i)\ \forall\ a_i \in B\} \quad (3)$$

Una relación de inseparabilidad (*indiscernibility relation*) que sea definida a partir de formar subconjuntos de elementos de U que tienen igual valor para un subconjunto de atributos B de A, $B \subseteq A$, es una relación de equivalencia.

Los conceptos básicos de la RST son las Aproximaciones Inferiores y Superiores de un subconjunto $X \subseteq U$ (Pawlak 1992). Estos conceptos fueron originalmente introducidos con referencia a una relación de inseparabilidad R. Sea R una relación binaria definida sobre U la cual representa la inseparabilidad, se dice que $R(x)$ significa el conjunto de objetos los cuales son inseparables de x. Así, $R(x) = \{y \in U: yRx\}$.

En la RST clásica, R es definida como una relación de equivalencia; es decir, es una relación binaria $R \subseteq UxU$ que es reflexiva, simétrica y transitiva. R induce una partición de U en clases de equivalencia correspondiente a $R(x)$, $x \in U$.

Este enfoque clásico de RST es extendido mediante la aceptación que objetos que no son inseparables pero sí suficientemente cercanos o similares puedan ser agrupados en la misma clase (Slowinski and Vanderpooten 1997). El objetivo es construir una relación R' a partir de la relación de inseparabilidad R pero flexibilizando las condiciones originales para la inseparabilidad. Esta flexibilización puede realizarse de múltiples formas, así como pueden darse varias definiciones posibles de similitud. Existen varias funciones de comparación de atributos (funciones de similitud), las cuales están asociadas al tipo del atributo que se compara (Wilson and Martínez 1997). Sin embargo, la relación R' debe satisfacer algunos requerimientos mínimos.

Si R es una relación de inseparabilidad definida en U, R' es una relación de similitud extendida de R si y solo si $\forall x \in U$, $R(x) \subseteq R'(x)$ y $\forall x \in U$, $\forall y \in \boldsymbol{R'(\,)\,x}$, $R(y) \subseteq \boldsymbol{R'(\,)\,x}$, donde $R'(x)$ es la clase de similitud de x, es decir, $R'(x) = \{y \in U: yR'x\}$. R' es reflexiva, cualquier clase de similitud puede ser vista como un agrupamiento de clases de inseparabilidad y R' induce un cubrimiento de U (Skowron and Stepaniuk 1992). Esto muestra que un objeto puede pertenecer a diferentes clases de similitud simultáneamente, lo que significa que la granulación inducida por R' sobre U no es necesariamente una partición, sino un cubrimiento.

La aproximación de un conjunto $X \subseteq U$, usando una relación de inseparabilidad R, ha sido inducida como un par de conjuntos llamados **aproximaciones *R*-inferior y *R*-superior de *X*** (Bazan, Son et al. 2003). Se considera en esta tesis una definición de aproximaciones más general, la cual maneja cualquier relación reflexiva *R'*. Las aproximaciones *R'*-inferior ($R'_*(X)$) y *R'*-superior ($R'^*(X)$) de X están definidas respectivamente como se muestra en las expresiones (4) y (5).

$$R'_*(X) = \{x \in X : R'(x) \subseteq X\} \tag{4}$$

$$R'^*(X) = \bigcup_{x \in X} R'(x) \tag{5}$$

Teniendo en cuenta las expresiones definidas en (4) y (5), se define la **región límite** de X para la relación *R'* (Deogun 1995):

$$BN_b(X) = R'^*(X) - R'_*(X) \tag{6}$$

Si el conjunto BN_B es vacío entonces el conjunto X es exacto respecto a la relación *R'*. En caso contrario, $BN_B(X) \neq \varnothing$, el conjunto X es inexacto o aproximado con respecto a *R'*.

Usando las aproximaciones inferior y superior de un concepto X se definen tres regiones para caracterizar el espacio de aproximación: la región positiva que es la aproximación *R'*-inferior, la región límite que es el conjunto BN_B y la región negativa ($NEG(X)$) que es la diferencia entre el universo y la aproximación *R'*-superior. Los conjuntos $R'_*(X)$ (denotado también como $POS(X)$), $R'^*(X)$, $BN_B(X)$ *y* $NEG(X)$ son las nociones principales de la Teoría de Conjuntos Aproximados.

1.3.2 Medidas de inferencia clásicas de la Teoría de los Conjuntos Aproximados

La Teoría de los Conjuntos Aproximados ofrece algunas medidas para analizar los sistemas de información (Skowron 1999; Arco, Bello et al. 2006). A continuación se muestran las principales. En las expresiones 7 a la 10 se emplean las aproximaciones R'-inferior (R'_*)

(X) y R'-superior (R'*) (X) de X, las cuales están definidas en las expresiones (4) y (5) respectivamente.

Precisión de la aproximación. Un conjunto aproximado X puede caracterizarse numéricamente por el coeficiente llamado precisión de la aproximación, donde |X| denota la cardinalidad de X, X≠ ∅. Observe la expresión (7).

$$\alpha(\mathrm{X}) = \frac{|\mathrm{R'}_*(\mathrm{X})|}{|\mathrm{R'}^*(\mathrm{X})|} \tag{7}$$

Obviamente, $0 \leq \alpha(X) \leq 1$. Si $\alpha(X)=1$, X es duro (exacto), si $\alpha(X) < 1$ X es aproximado (vago, inexacto), siempre respecto al conjunto de atributos considerado (Skowron 1999).

Calidad de la aproximación. El coeficiente siguiente

$$\gamma(X) = \frac{|R'_*(X)|}{|X|} \tag{8}$$

expresa la proporción de objetos que pueden ser correctamente clasificados en la clase X. Además, $0 \leq \alpha(X) \leq \gamma(X) \leq 1$, y $\gamma(X)=0$ si y solo si $\alpha(X)=0$, mientras $\gamma(X)=1$ si y solo si $\alpha(X)=1$.

Considerando que $X_1,\ldots, X_l$ son las clases del sistema de decisión, se define la medida:

Calidad de la clasificación. Este coeficiente describe la inexactitud de las clasificaciones aproximadas:

$$\gamma(X) = \frac{|\sum_{i=1}^{n} R'_*(X_i)|}{|U|} \tag{9}$$

La medida calidad de la clasificación expresa la proporción de objetos que pueden clasificarse correctamente en el sistema. Si ese coeficiente es igual a 1, entonces el sistema de decisión es consistente, en otro caso es inconsistente (Filiberto *et al.*, 2011).

Función de pertenencia aproximada. Esta función cuantifica el grado de solapamiento relativo entre $R'(x)$ (clase de similitud de x) y la clase a la cual el objeto x pertenece. Se define como sigue:

$$\mu_x(x) = \frac{|X \cap R'(x)|}{|R'(x)|} \quad (10)$$

La función de pertenencia aproximada puede ser interpretada como una estimación basada en frecuencias de $Pr\,(x \in X \mid x, R'(x))$, es decir, la probabilidad condicional de que el objeto x pertenezca al conjunto X (Grabowski 2003).

1.4 Reducción de datos

Se abordaran la reducción de datos desde la perspectiva de reducir la dimensionalidad del problema y con ello simplificar el clasificador o la función de minería de datos a emplear. Con esto se incrementa la velocidad de manipulación de los datos, y mejora el desempeño reduciendo la influencia de los ruidos, de ahí su importancia en el preprocesamiento de los datos.

1.4.1 Selección de rasgos

La Selección de rasgos reduce la dimensionalidad del problema, se dice que es el proceso de búsqueda a través del conjunto A de n rasgos para tratar de encontrar el mejor subconjunto entre los 2^n-1 subconjuntos candidatos de acuerdo a alguna medida de evaluación, por esta razón tenemos dos componentes:

- Una función de evaluación usada para evaluar un subconjunto de rasgos, en cuanto a la calidad del mismo o su capacidad discriminante. Son ejemplos de función de evaluación: distancia entre clases, dependencia probabilística, medida de información (Ej. Entropía), medida de consistencia u otras medidas de la RST.

- Un Procedimiento de Generación usado para generar subconjuntos de rasgos candidatos.

El procedimiento para la generación puede ser una estrategia de búsqueda exhaustiva, a ciegas o heurística (Russell, Norvig, 2016), en este último caso pueden utilizarse metaheurísticas como Algoritmos Genéticos (Goldberg, 1989), PSO (Kennedy, 2011), Colonia de hormiga (Dorigo, Birattari, Stützle, 2010). Métodos como Focus; búsqueda exhaustiva, ABB: búsqueda completa, SetCover: búsqueda heurística, LVF: búsqueda

probabilística, QBB: búsqueda híbrida se implementan en WEKA (Witten, I, Frank, E., Hall M., Pal, C., 2017) como procedimientos para la generación de candidatos.

Dependiendo de quién es la función de evaluación los selectores pueden ser de dos tipos: Filter (si la función de evaluación es independiente del algoritmo inductivo) y Wrapper (usa el algoritmo inductivo como función de evaluación).

Desde RST también se analiza este problema basado en el concepto de reductos. Según (Caballero 2005), un reducto es un conjunto reducido de atributos que preserva la partición del universo. El uso de reductos en la selección y reducción de atributos ha sido ampliamente estudiado (Pal and Skowron 1999; Ahn 2000). El algoritmo QuickReduct es un ejemplo de método heurístico para construir reductos, el mismo construye un subconjunto de rasgos que tiene igual valor de la medida calidad de la clasificación que el valor de esta medida si se utilizan todos los rasgos.

Algoritmo QuickReduct

```
Dado S= (U, A) donde |U|=n, |A|= m.
P1: R= {}, A*=A, FIN= false.
P2: Repeat
       P2.1 Seleccionar Ai de A*: f(R∪{Ai}) es  Máximo.
       P2.2 Eliminar de A* el Ai seleccionado.
       FIN=true If Y(R) =Y (A)
      until FIN.
```

Selección de rasgos con Algoritmos Genéticos

Los Algoritmos Genéticos (AG[1]) surgen como herramientas para la solución de complejos problemas de búsqueda y optimización, producto del análisis de los sistemas adaptativos en la naturaleza, y como resultado de abstraer la esencia de su funcionamiento. Son un ejemplo de método que explota la búsqueda aleatoria "guiada" que ha ganado popularidad en los últimos años debido a la posibilidad de aplicarlos en una gran gama de campos y a las pocas exigencias que impone al problema.

Los AG trabajan a partir de una población inicial de estructuras artificiales que van modificando repetidamente a través de la aplicación de los siguientes operadores genéticos:

[1] En el texto se utilizará AG tanto para el plural como para el singular.

Operador de Selección o Darwiniano, Operador de Cruzamiento o Mendeliano, Operador de Mutación (Wróblewski, 1998).

Para utilizar los AG es necesario encontrar una posible estructura para representar las soluciones. Pensando este asunto como el problema de buscar en un espacio de estados, una instancia de esta estructura representa un punto o un estado en el espacio de búsqueda de todas las posibles soluciones. Así, una estructura de datos en el AG consistirá en uno o más cromosomas, el cual se representa comúnmente como una cadena de bits (existen otras representaciones) (Wroblewski, 1995).

Cada cromosoma (cadena) es una concatenación de un número de subcomponentes llamados genes. La posición de un gen en el cromosoma se conoce como locus y sus valores como alelos. En la representación como cadena de bits, un gen es un bit o una cadena de bits, un locus es su posición en la cadena y un alelo es su valor (0 ó 1 si es un bit). El AG ejecuta para un número fijo de generaciones o hasta que se satisface algún criterio de parada.

Uno de los problemas en los que se utiliza el AG es en el cálculo de reductos debido a la complejidad de este. A continuación se propone la solución al cálculo de reductos usando AG (Yang and Honavar, 1998).

Cada individuo es un posible reducto o un subconjunto de atributos. Los individuos se representan como cadenas de longitud n (cantidad de atributos). En estas cadenas si la posición i-ésima tiene valor 1 entonces, el atributo i-ésimo está en el subconjunto denotado por la cadena, en otro caso tiene valor 0. La función de calidad depende de: la cantidad de unos en el cromosoma y la cantidad de pares de objetos (con diferentes valores de decisión) separados por los atributos que están en el subconjunto representado por el individuo o cromosoma.

Selección de rasgos PSO

Otra metaheurística que puede utilizarse en la selección de rasgos es la Optimización basada en partículas (Particle Swarm Optimization, PSO), la cual consiste en la optimización basada en nubes de partículas, conjunto de técnicas inspiradas en el comportamiento de las bandadas de aves o bancos de peces.

Cada pájaro (partícula) es tratado como un punto en un espacio N dimensional el cual ajusta su propio "vuelo" de acuerdo a su propia experiencia y la experiencia del resto de la banda. La banda (swarm) vuela por el espacio buscando regiones prometedoras. Las partículas pueden ser simples agentes que vuelan a través del espacio de búsqueda y almacenan (y posiblemente comunican) la mejor solución que han descubierto. En PSO, las partículas nunca mueren. Cada partícula tiene una posición y velocidad en el espacio de búsqueda. Tiene una medida de calidad parecida a la "fitness" de un individuo, ellas "aprenden" ajustando su posición y velocidad, teniendo en cuenta su mejor posición (*best solution*) hasta el momento.

PSO se inicializa con una población de partículas. Cada partícula se trata como un punto en un espacio tridimensional. La partícula *i* se representa como $X_i = (x_{i1}, x_{i2},..., x_{is})$. La mejor anterior posición (*pbest,* la posición que da el mejor valor de aptitud) de cualquier partícula es $P_i = (p_{i1}, p_{i2},..., p_{is})$. El índice de los mejores globales de partícula está representada por '*gbest*'. La velocidad para la partícula *i* es $V_i = (v_{i1}, v_{i2},..., v_{iS})$. Las partículas son manipuladas según a la siguiente ecuación:

$$v_i = w * v_{id} + c_1 * rand(\quad) * (p_{id} - x_{id}) + c_2 * rand(\quad) * (p_{gd} - x_{id}) \qquad (11)$$

Donde *w* es el peso de inercia, selección adecuada de la inercia. El peso proporciona un equilibrio entre la exploración global y local y, por lo tanto, requiere menos iteraciones en promedio para encontrar el óptimo.

$$w = w_{max} - \frac{w_{max} - w_{min}}{iter_{max}} * iter \qquad (12)$$

Donde *Wmax* es el valor inicial del coeficiente de ponderación, *Wmin* el valor final del coeficiente de ponderación, *itermax* el número máximo de iteraciones o generaciones, e *iter* es la iteración actual o número de generación. Las constantes de aceleración *c1* y *c2* en (11) representan la ponderación de los términos de aceleración estocástica que tiran de cada partícula hacia *pbest* y las mejores posiciones; *rand ()* función aleatorias en el rango [0, 1]. Medida de dependencia entre atributos.

Un importante aspecto en el análisis de datos es descubrir dependencias entre atributos.

Dependencia de un atributo

Sean *B* y *D* subconjuntos del conjunto *A* en el sistema de información (*U*, *A*). Un conjunto de atributos *D* depende totalmente de un conjunto de atributos *B* denotado por $B \Longrightarrow D$, si todos los valores de los atributos de *D* son unívocamente determinados por los valores de los atributos en *B*, si existe una dependencia funcional entre los valores de *D* y *B*.

Se dice que *D* depende de *B* en un grado *k* ($0 \leq k \leq 1$), denotado por $B \Rightarrow_k D$, para el valor de *k* definido por la expresión (13).

$$k = \sigma(B, D) = \frac{|Pos_B(D)|}{|U|} \tag{13}$$

donde

$$Pos_B(D) = \bigcup_{x \in U/_D} B_*(X) \tag{14}$$

Si k=1 entonces se dice que *D* depende totalmente de *B*, mientras que si $k<1$ se dice que *D* depende parcialmente de *B*. (Bello, García and Pérez, 2012)

Peso de un atributo

La importancia de un atributo *a* en la tabla de decisión *S = (U, C∪D)*, donde *U* es el universo de objetos, *C* es el conjunto de atributos de condición y *D* es el conjunto de atributos de decisión, se puede evaluar midiendo el efecto de remover a de *C*.

Como se definió en la expresión (13), $\sigma(C, D)$ expresa el grado de dependencia entre los conjuntos de atributos *C* y *D*. Cuando se modifica *C* o *D*, este valor puede variar, entonces la diferencia entre $\sigma(C, D)$ y $\sigma(C - \{a\}, D)$ da una medida del grado en que el atributo *a* influye en $\sigma(C, D)$. A partir de la normalización de esta diferencia se puede calcular el peso del atributo a según la expresión (14).

$$w_{(C,D)}(a) = \frac{(\sigma(C, D) - \sigma(C - \{a\}, D))}{\sigma(C, D)} = 1 - \frac{\sigma(C - \{a\}, D)}{\sigma(C, D)} \tag{15}$$

El coeficiente w(a) se puede interpretar como el error que ocurre cuando el atributo a es removido del sistema.

1.4.2 Selección de ejemplos

La selección de los objetos de un dominio a incluir en un conjunto de entrenamiento es un problema presente en todos los modelos computacionales que realizan inferencias a partir de ejemplos. El mismo se conoce como edición de conjuntos de entrenamiento. La edición se hace con el objetivo de eliminar los prototipos que inducen a una incorrecta clasificación supervisada, seleccionando un conjunto de referencia representativo y reducido. Las técnicas de edición, aunque también producen la eliminación de prototipos, y con ello, la reducción de la matriz de aprendizaje, tienen como objetivo fundamental el obtener una muestra de entrenamiento de mejor calidad para una mejor precisión del sistema. Así, además de conseguir una reducción (en ocasiones notables) del conjunto de referencia, ellas proporcionan un conjunto de "calidad" que hace incrementar la precisión de los métodos de clasificación supervisada cuando se aplican sobre conjuntos editados y decrecer a su vez el costo computacional de dicha clasificación.

Según (García, Villuendas et al. 2005) los métodos de edición se pudieran agrupar en: i) Métodos de selección de objetos (aquellos en los cuales solo se seleccionan objetos del conjunto de entrenamiento siguiendo cierto criterio) y ii) Métodos de construcción de objetos (aquellos en los que los nuevos prototipos pueden no estar en la matriz de entrenamiento).

En la Teoría de los Conjuntos Aproximados resulta de gran interés el significado de la Aproximación Inferior de un sistema de decisión. En la Aproximación Inferior de cada clase de un sistema se agrupan aquellos objetos que con absoluta certeza pertenecen a su clase. Esto garantiza que los objetos contenidos en la Aproximación Inferior están exentos de ruidos.

La idea básica de aplicar los conjuntos aproximados para editar los conjuntos de entrenamiento es la siguiente: en el conjunto de entrenamiento se colocan los objetos del sistema de decisión inicial que pertenecen a la aproximación inferior de cada clase. Esto es igual a decir que el conjunto de entrenamiento resultante de este primer algoritmo será la región positiva del sistema de decisión. En esta forma, los objetos que están etiquetados incorrectamente o muy cerca de la frontera de decisión pueden eliminarse del conjunto de entrenamiento, los cuales afectan la calidad de la inferencia. Como en la aproximación inferior de cada clase estarán aquellos objetos que con certeza pertenecen a dicha clase, se

garantiza la eliminación de cualquier presencia de ruido en el sistema de decisión (Bello, García and Pérez, 2012).

Algoritmo Edit1RS:

P1. Construir el subconjunto B, $B \subseteq A$. Se sugiere que B sea un reducto del sistema de decisión para disminuir la dimensionalidad de los atributos sin afectar el proceso de clasificación.
P2. Formar los conjuntos $X_i \subseteq U$, tal que todos los elementos del universo (U) que tienen valores d_i en el atributo de decisión están en X_i.

P3. Para cada conjunto X_i, se calcula su aproximación inferior ($R'_*(X_i)$) (descrita en la expresión 1.4) respecto al subconjunto B de atributos construido en el paso P1 y la relación R'.
P4. Construir el conjunto de entrenamiento editado S_E como la unión de todos los conjuntos $R'_*(X_i)$.

En Edit1RS se toman en cuenta sólo los elementos que están en las aproximaciones inferiores. Sin embargo, la información de los objetos que están en la frontera (BN_B), pueden ser interesantes, pero habría que cambiar la clase a aquellos que lo requieran si se quieren eliminar de la frontera, esto pudiera hacerse a través de un proceso de re-etiquetamiento.

La función de pertenencia aproximada, descrita en la expresión (10), arroja un valor de pertenencia de un objeto a una clase determinada. Si se le aplica esta función a cada uno de los objetos del conjunto frontera para cada una de las clases, se puede saber por cada objeto, a cuál de las clases él tiene mayor grado de pertenencia. Esto constituye una forma de re-etiquetar las clases de los objetos del conjunto frontera (Caballero, Bello et al., 2006b) .El algoritmo Edit3RS se propuso tomando en cuenta estas ideas.

Algoritmo Edit3RS:

P1. Construir el conjunto *B*, $B \subseteq A$. Se sugiere que *B* sea un reducto del sistema de decisión para disminuir la

dimensionalidad de los atributos sin afectar el proceso de clasificación.

P2. Formar los conjuntos $X_i \subseteq U$, tal que todos los elementos del universo (U) que tienen valores d_i en el atributo de decisión están en X_i.

P3. $S_E = \varphi$.

P4. Para cada conjunto X_i:

Calcular su aproximación inferior ($R'_*(X_i)$) (descrita en la expresión 1.4) y su aproximación superior ($R'^*(X_i)$) (descrita en la expresión 1.5), respecto al subconjunto B de atributos construido en el paso P1 y la relación R'.

p4.1. $S_E = S_E \cup R'_*(X_i)$. **p4.2.** $T_i = R'^*(X_i) - R'_*(X_i)$.

P5. Calcular la unión de los conjuntos T_i, para obtener $T = \cup T_i$

P6. Para cada elemento x del conjunto T:

p6.1. Calcular la función de pertenencia aproximada a cada clase X, según la expresión (1.10)

p6.2. Si la clase con mayor valor (según p6.1) difiere de la clase del objeto, entonces reetiquetarla por esta clase con la cual se obtuvo mayor valor de pertenencia del objeto.

P7. $S_E = S_E \cup T$. El conjunto de entrenamiento editado se obtiene como el conjunto resultante en S_E.

El método Edit1RS es un método de selección de objetos (selecciona aquellos que están en las aproximaciones inferiores de las clases) y el método Edit3RS es un método de construcción de objetos, pues no solo selecciona objetos, sino que puede obtener nuevos a partir del proceso de re-etiquetamiento que se aplica a los objetos de la frontera, donde puede ser que algunos de estos cambien su clase (Caballero, 2007).

En este trabajo se propone una modificación al algoritmo Edit1RS, permitiendo un cierto grado de inconsistencia en los conjuntos editados.

1.5 Extensiones de la RST

En el caso de la construcción de conjuntos aproximados cuando se tienen rasgos de dominios continuos en lugar de discretos, y no se quiere discretizar los datos, es necesario usar relaciones de similaridad en lugar de relaciones de equivalencias.

La diferencia está en que las relaciones de equivalencias son relaciones reflexivas, simétricas y transitivas, mientras que las relaciones de similaridad no tienen alguna de estas propiedades; usualmente son relaciones reflexivas y simétricas, no transitivas.

Las relaciones de similaridad generan una granulación del universo que es un cubrimiento, no una partición como lo hace las relaciones de equivalencias. Un cubrimiento es una granulación compuesta por gránulos, que unidos dan el universo pero que la intersección de ellos no tiene que ser vacía; es decir, un objeto pertenece a dos o más gránulos.

Una relación de similaridad puede enunciarse de la forma siguiente: x R y si y solo si F(x,y) $\geq \varepsilon$, donde e es un umbral y F(x,y) es una función de semejanza como la de la expresión (15) (Filiberto *et al.*, 2011).

$$F(x,y) = \sum_{i=1}^{n} w_i * \partial_i(x_i, y_i) \tag{15}$$

Donde ∂i (x_i,y_i) es una función de comparación para comparar los valores del rasgo *i*. Se pueden usar muchas funciones d*i*, pero. En este trabajo se propone usar [16]:

$$\partial_i(x_i, y_i) = \begin{cases} 1 - \dfrac{|x_i - y_i|}{Max(a_i) - Max(a_i)}, si \;\; i \; es \; continuo \\ 1 \quad , si \; i \; discreto \;\; y \; x_i = y_i \\ 0 \quad , si \; i \; discreto \;\; y \; x_i \neq y_i \end{cases} \tag{16}$$

En este caso se pueden usar a la vez rasgos con dominios discretos y continuos. Los valores w_i son los pesos de los rasgos, los cuales se dan por datos o se le puede asignar el valor 1/n a cada uno, cuando todos los rasgos tienen la misma importancia.

Cuando solo se tengan rasgos continuos también se pueden usar relaciones como las siguientes, donde se emplea la distancia euclidiana: x R y si y solo si d(x,y)$\leq \varepsilon$, donde ε es un umbral.

$$d_e(x,y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{17}$$

Usando relaciones como las anteriores se pueden definir las clases de similaridad siguientes:

$R(x)= \{ y \in U : yRx \}$, objetos similares a x

$R^{-}(x)= \{ y \in U : xRy \}$, objetos a los que x es similar

Un ejemplo de una extensión de la RST basada en relaciones de semejanza fue presentada por R. Slowinski y D. Vanderpooten (Slowinski and Vanderpooten 2000). Ellos definen las aproximaciones inferior y superior de un conjunto X de la forma siguiente.

Sean $X \subseteq U$ y R una relación binaria y reflexiva sobre U. Entonces la aproximación inferior y superior se define como se muestra en las expresiones (18) y (19) respectivamente, donde R^{-1} denota la relación inversa de R (Slowinski, Vanderpooten, 2000).

$$R'_{*}(X) = \{x \in X: R^{-}(x) \subseteq X\} \quad (18)$$

$$R'^{*}(X) = \{x \in X: R^{-}(x) \cap X \neq \emptyset\} \quad (19)$$

1.6 *Big Data* en el análisis inteligente de los datos

La gran cantidad de datos disponibles en la actualidad junto con las herramientas necesarias para su procesamiento conforman lo que hoy en día conocemos como *Big Data*. El volumen actual de datos supera las capacidades de procesamiento de los sistemas clásicos de minería de datos (Malvicino and Yoguel, 2016).

La era del *Big Data* o datos masivos, se caracteriza por la presencia de gran volumen, velocidad y variedad. Este término no sólo se refiere al tamaño de los conjuntos de datos a manejar, sino que también implica la necesidad de su análisis en tiempo real cuando recibimos flujos de datos, y la posibilidad de encontrarnos con una gran variedad de estructuras, datos numéricos, textuales, enlaces, etc. Se trata con datos cuyo volumen, diversidad y complejidad requieren el uso de nuevas arquitecturas, técnicas, algoritmos y análisis para gestionar y extraer el valor y conocimiento oculto en ellos por ello se habla de las 3 V (Mayer-Schönberger and Cukier, 2013).

Existen diversas herramientas diseñadas por los gigantes tecnológicos. La aproximación más popular es el sistema de procesamiento distribuido MapReduce (Dean and Ghemawat, 2008),

presentado por Google en 2003. A partir de ahí, se han desarrollado aplicaciones y tecnologías que siguen su estela o exploran otras estrategias de procesamiento. Posteriormente, junto con otras grandes firmas, se creó la Fundación Apache que desarrolla más de 150 proyectos de software libre para *big data*, apadrinando el proyecto Spark, el cual constituye el nuevo referente para el diseño de algoritmos para el procesamiento de datos masivos (Herrera 2014).

Si se tiene que procesar un conjunto de datos del tamaño de un terabyte en un ordenador que es capaz de procesar 25 megabytes por segundo. El tiempo total de procesamiento sería aproximadamente de 46 días. En cambio sí se dispone de un clúster de 1.000 ordenadores y podemos paralelizar mediante el procesamiento distribuido clásico, el tiempo de cálculo se reduciría a 66 minutos. De ahí, la necesidad de un nuevo paradigma para de análisis de datos.

Apache Spark es un sistema para el procesamiento de datos masivos, centrado en la velocidad por el procesamiento en memoria, facilidad de uso y un análisis sofisticado para el diseño de algoritmos (Salloum *et al.*, 2016). Es posible escribir una aplicación completa en Python, Java o Scala compilarla y ejecutarla en el clúster, y ofrece aplicaciones para la realización de diferentes tipos de análisis: explotar los datos con un lenguaje SQL (Spark SQL) (Armbrust *et al.*, 2015), procesamiento de flujo continuo de datos (Spark Streaming), una librería de aprendizaje automático (MLlib) y una aplicación para el procesamiento de grafos (GraphX).

Python es ideal para trabajar con grandes volúmenes de datos porque favorece su extracción y procesamiento, siendo el elegido por las empresas de *Big Data*. A nivel científico, posee una amplia biblioteca de recursos con especial énfasis en las matemáticas para aspirantes a programadores en áreas especializadas. Es un lenguaje de programación poderoso y fácil de aprender. Cuenta con estructuras de datos eficientes, de alto nivel y un enfoque simple pero efectivo para la programación orientada a objetos. La elegante sintaxis de Python y su tipado dinámico, junto con su naturaleza interpretada, hacen de éste un lenguaje ideal para *scripting* y desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas.

Para análisis inteligente de datos se reportan el scikit-roughsets: paquete implementado en Python basado en RST que implementa la relaciones de equivalencia, el cálculo de las aproximaciones inferiores basadas en estas relaciones de equivalencia y el cálculo de reducto pero de forma exhaustiva.

1.7 Conclusiones parciales

Los grandes volúmenes de datos existentes en la actualidad requieren procesamiento para convertirse en conocimiento.

El preprocesamiento de datos engloba a todas aquellas técnicas de análisis de datos que permiten mejorar la calidad de ellos e incluye los procesos de recopilación de datos, limpieza, transformación y reducción, y no siempre se aplican en un mismo orden.

La Teoría de los Conjuntos Aproximados (RST) y sus extensiones constituyen una herramienta para el análisis de datos, de gran utilidad para encontrar conjuntos reducidos de atributos y editar conjuntos de entrenamiento.

Python constituye un lenguaje de programación muy utilizado por la comunidad del aprendizaje automatizado muy factible para implementar los principales métodos computacionales y conceptos para el análisis de datos, es ideal para trabajar con grandes volúmenes de datos porque favorece su extracción y procesamiento

Para implementaciones computacionales en *Big Data*, Apache Spark resulta el sistema recomendado actualmente.

Capítulo 2

Diseño e implementación de una biblioteca en Python para el preprocesamiento de datos basado en RST

2. DISEÑO E IMLEMENTACIÓN DE UNA BIBLIOTECA EN PYTHON PARA EL PREPROCESAMIENTO DE DATOS BASADO EN LA RST

A continuación se describen los distintos paquetes implementados y la validación de los resultados alcanzados para los paquetes de selección de rasgos y edición de conjuntos de entrenamientos.

2.1 Caracterización de la biblioteca

Se desarrolló la biblioteca RoughPython en el lenguaje Python para el preprocesamiento de datos usando la RST y sus extensiones. La biblioteca está formada por tres paquetes principales: RSTBasic donde están implementadas los conceptos básicos de esta teoría, cálculo de relaciones de equivalencia y similitud, medidas y las definiciones fundamentales de RST; Aproximación Inferior y Aproximación Superior.

Para el preprocesamiento como parte de la reducción de datos se implementaron dos paquetes uno para la selección de rasgos y otro para la selección o edición de ejemplos.

En el paquete de selección de ejemplos RSTEdit se implementaron los algoritmos Edit1RS, Edit2RS, y Edit3RS que constituyen métodos de construcción y selección de prototipos. En ambos se utilizan las funciones que permiten obtener las aproximaciones inferior y superior del conjunto de entrenamiento referido y que fueron implementadas en el paquete RSTBasic.

El paquete de selección de rasgos RSTFeatureSeleccion contiene el algoritmo QuickReduct para la selección de rasgos basados en los principales conceptos y definiciones de la RST, el cual utiliza como método de búsqueda el método ascensión de colina (hill-climbing). Este paquete presenta además la solución al problema de cálculo de reductos utilizando las metaheurísticas Algoritmos Genéticos (AG) y Optimización Enjambre de Partículas (PSO). En la figura 2.1 se presenta un esquema donde se presenta la estructura de la biblioteca RoughPython.

Además este paquete tiene las bases de casos que se necesiten para evaluar los resultados.

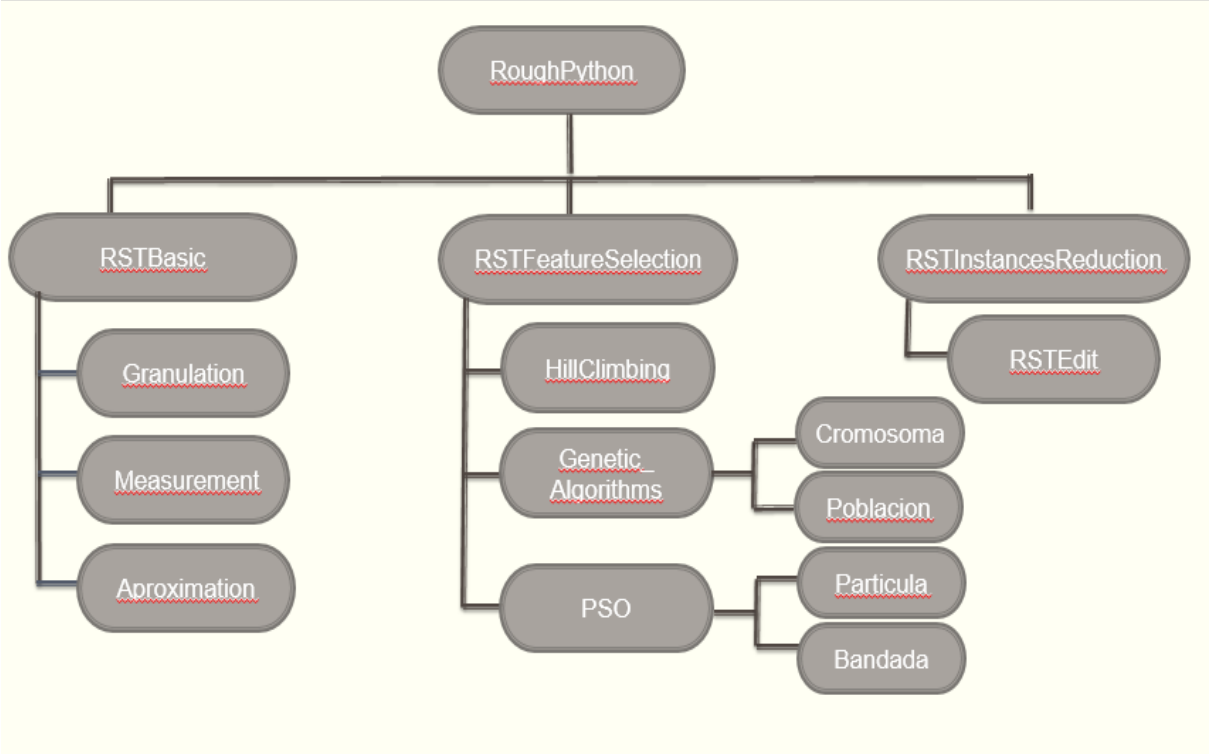

Figura 2.1 Esquema de paquetes de la biblioteca RoughPython

2.2 Paquete RSTBasic

En este paquete se conciben tres ficheros básicos Granulation.py, Measurement.py y Aproximation.py donde están implementados las definiciones, medidas y conceptos básicos de la RST y sus extensiones.

En el fichero Granulation.py se implementan las relaciones de inseparabilidad (*indiscernibility relation*) definidas en el epígrafe 1.3.1, donde un par de objetos son inseparables de acuerdo a una relación que puede ser de equivalencia o de similitud. En este paquete se utilizan funciones auxiliares que permiten leer las bases de casos con formato arff, para ser procesadas y poder utilizar la información de estas. Un fichero formato arff relaciona el nombre de la base de casos, declara los atributos y la data (Witten, I, Frank, E., Hall M., Pal, C., 2017).

En el fichero Measurement.py se implementan las medidas de inferencia como Precisión de la aproximación (ecuación 7), Calidad de la aproximación (ecuación 8), Función de pertenencia aproximada (ver ecuación 10) y Calidad de la clasificación (ecuación 9), así como medidas para la dependencia y peso de los atributos (ecuación 13 y 15

respectivamente). Estas medidas y las funciones que calculan las aproximaciones inferior y superior implementadas en Aproximation.py utilizan las clases que se obtengan al aplicar relación de equivalencia o de similitud y las clases de decisión.

2.3 Paquete RSTFeatureSelection

En este paquete se implementan distintos métodos de selección de rasgos, para ello se busca a partir de un conjunto de n rasgos los subconjuntos de rasgos candidatos. La función de evaluación usada para evaluar un subconjunto de rasgos, en cuanto a la calidad del mismo es la medida de calidad de clasificación o medida de consistencia (ecuación 9) y como procedimiento de generación se utilizan búsqueda heurística (hill-climbing) implementada por el método `quickReduct(clases, list_rasgos)` y las clases Poblacion y Cromosoma que implementa la metaheurística Algoritmos Genéticos (AG) y Optimización Enjambre de Partículas (PSO) representada por las clases Bandada y Particula.

En la selección de rasgos aplicando búsqueda heurística se utilizó la selección basada en principios y conceptos de la RST a partir del algoritmo QuickReduct (epígrafe 1.4.1), el método `quickReduct(clases, list_rasgos)` implementa este algoritmo, se le pasan como parámetros las clases de decisión y los atributos.

2.3.1 Implementación usando AG

El cálculo de reductos a partir de AG se implementó de la siguiente forma: cada individuo o cromosoma es un posible reducto. Se crea una clase Cromosoma, en la cual los individuos se representan como cadenas de longitud n, donde n representa la cantidad de atributos, en estas cadenas si la posición i-ésima tiene valor 1 entonces, el atributo i-ésimo está en el subconjunto denotado por la cadena, en otro caso tiene valor 0. La primera generación de cromosomas se genera de forma aleatoria.

Esta clase Cromosoma incluye la función de calidad, implementada mediante la `funcion_de_evaluacion (granulacion)` la cual recibe como parámetro las clases de decisión porque se utiliza la medida de calidad de la clasificación para evaluar el reducto `y` además la operación de mutación `mutar ()`.

La clase Poblacion está formada por una lista de cromosomas, en esta clase están implementadas las operaciones selección y cruce y funciones para la evolución y generación de nuevos conjuntos de cromosomas. En esta clase también se almacenan los mejores individuos para obtener el mejor reducto en una cantidad de iteraciones.

El diagrama de las clases fundamentales para implementar AG se muestra en la figura 2.2.

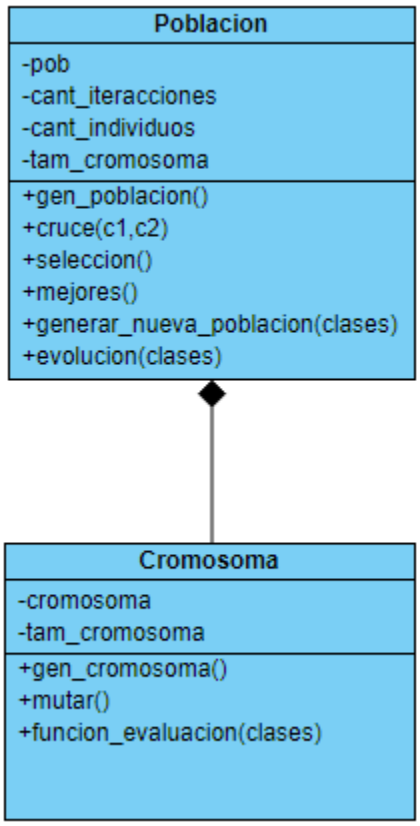

Figura 2.2 Diagrama de clases para AG

Para realizar el cálculo de reductos usando AG es necesario crear una Población con sus atributos necesarios: cantidad de individuos por población, tamaño del cromosoma o conjunto de rasgos y cantidad de iteraciones, y la nueva población se crea llamando a la función evolución que es la que permite el proceso de generación de poblaciones.

2.3.2 Implementación usando PSO

La bandada (swarm) vuela por el espacio en la búsqueda de regiones prometedoras. Cada partícula tiene una posición y velocidad en el espacio de búsqueda. En el caso del cálculo de

reductos usando PSO se crearon dos clases Bandada y Partícula. La figura 2.4 muestra el diagrama de clases que implementa el cálculo de reductos utilizando PSO.

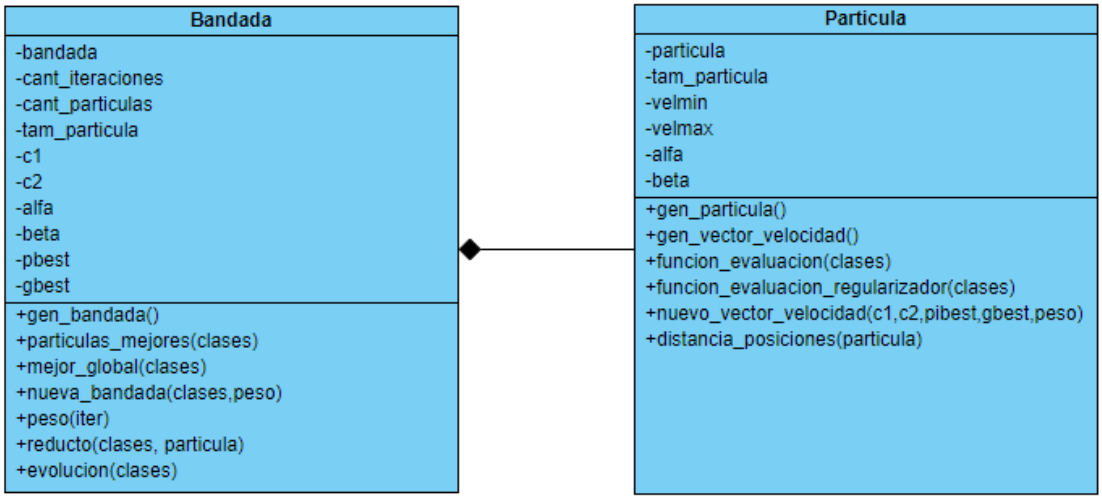

Figura 2.3 Diagrama de Clases PSO

Cada Partícula se representa por una cadena de n caracteres, donde n es la cantidad de rasgos o atributos. Además se implementa la función de calidad o *fitness*, a partir de la medida de calidad de la clasificación (ecuación 9) para medir la consistencia del sistema de decisión. Esta medida expresa que si *S* es un conjunto de rasgos consistentes, no existen dos instancias con los mismos valores en *S* que pertenecen a clases diferentes, de esta forma la medida no maximiza la separabilidad entre clases, sino que intenta conservar la potencia discriminante del conjunto de rasgos original (Wang, Yang, Teng, Xia, & Jensen, 2007).

En la clase Bandada aparecen las funciones que permiten generar el conjunto de partículas o bandada quedándose con la mejor posición en cada momento (posición local) y obteniendo el mejor conjunto de partículas o reducto y la partícula mejor global. Es necesario crear una Bandada con sus atributos correspondientes para generar a partir de la función evolución la lista de reductos.

2.3.3 Regularización de la función de calidad

Para mejorar el subconjunto de rasgos resultantes (reducto) se utilizó un regularizador, con el cual no solo se tiene en cuenta calidad del conjunto de rasgos sino también se busca la menor cantidad de rasgos siempre y cuando la calidad del reducto sea igual a la calidad

considerando todos los rasgos del sistema de decisión original (Wang, Yang, Jensen, & Liu, 2006).

Se utiliza como función fitness:

$$fitness = \alpha * \gamma R(D) + \beta * \frac{|C| - |R|}{|C|} \quad (20)$$

Donde $\gamma R(D)$ medida de calidad de la clasificación, $|R|$el número de '1' en la partícula o número de atributos seleccionados y $|C|$ total de atributos, α y β son dos parámetros que corresponden a la importancia de la calidad de la clasificación y la longitud del subconjunto $\alpha \in [0,1]$ y $\beta = 1 - \alpha$. El objetivo es maximizar esta función fitness.

2.3.4 Discusión de los resultados

Para evaluar la eficacia de los métodos implementados para la selección de rasgos se utilizan cinco bases de casos del repositorio UCI (Bache, Lichman, 2013). La tabla 1 muestra la información relacionada con las bases de casos.

Base de casos	Cantidad de casos	Cantidad de atributos	Cantidad de clases
breast-cancer-wisconsin	699	10	2
heart-statlog	270	15	2
labor	57	17	2
hepatitis	155	19	2
diabetes	768	9	2
lung-cancer	32	56	3

Tabla 1. Información de las bases de casos utilizadas para evaluar la Selección de rasgos

La tabla 2 muestra los resultados de evaluar la selección usando PSO sin regularizador y con regularizador. Como se puede apreciar, en la mayoría de los casos el uso del regularizador genera reductos con un menor número de rasgos.

Bases de Casos	Cantidad de rasgos	Cantidad de iteraciones	Longitud promedio de los reductos sin regularizador	Longitud promedio de los reductos con regularizador
breast-cancer-wisconsin.arff	9	100	5.3442	4.7025
heart-statlog.arff	13	100	7.3540	6.3118
labor.arff	16	100	9.4784	7.5381
hepatitis.arff	19	100	9.5650	10.2337
diabetes.arff	8	100	3.8612	3.5681
lung-cancer.arff	56	100	29.7854	26.8843

Tabla 2. Resultados de evaluar PSO

De igual forma se realizó el análisis usando AG mostrándose similares resultados. Se muestra cómo se logra la reducción en la cantidad de rasgos y que el uso de un regularizador disminuye la cantidad de estos. Se analizan los resultados alcanzados con ambos métodos corroborados con el test Wilcoxon (Wilcoxon, 1945).

2.4 Paquete RSTInstancesReduction

En el paquete RSTInstancesReduction de selección de ejemplos se implementaron los algoritmos Edit1RS, Edit2RS, y Edit3RS que constituyen métodos de construcción y selección de prototipos. En ambos se utilizan las funciones que permiten obtener las aproximaciones inferior y superior del conjunto de entrenamiento referido y que fueron implementadas en el paquete anterior. En el fichero RSTEdit están implementados estos tres algoritmos.

La función `edit1RS (list_conjuntos_aproximar, list_rasgos)` edita los conjuntos a partir del algoritmo Edit1RS, el cual elimina todos los elementos que no estén en el conjunto de aproximaciones inferiores. Se tienen como parámetros de la función: el conjunto de las clases de decisión y la lista de los rasgos para obtener la granulación por condición.

La función `edit3RS (list_conjuntos_aproximar, list_rasgos)` implementa el algoritmo Edit3RS, pues este no elimina objetos sino que re-etiqueta los que pertenecen a las fronteras de las clases usando el grado de pertenencia de estos a las clases. Por lo que se necesita como parámetro las clases de decisión para calcular la medida de pertenencia de cada instancia a la clase decisión. La medida de pertenencia que se utiliza es la medida de la RST (ecuación 10).

2.4.1 Diseño e implementación del Edit2RS

En este trabajo se propone una modificación del método Edit1RS, denominado Edit2RS y en el que se cambia la forma de calcular la aproximación inferior de un objeto. Siendo esta aquella a la que pertenecen los objetos que tienen un grado de pertenencia al conjunto que se aproxima mayor que un umbral dado, cuando este umbral tiene valor 1 se tiene los mismos resultados que con la definición original; esta propuesta se basa en la definición de aproximaciones con enfoque probabilístico propuesta en (Pawlak, Wong, Ziarko, 1988), ver expresión (4):

$$B_*(X) = \{x \in U | \, \mu^{\alpha\beta} x(x) \geq \ \alpha\}$$

El método Edit2RS ofrece una variante intermedia entre los métodos Edit1RS y Edit3RS. Edit1RS solo considera casos completamente consistentes, Edit3RS considera todos los objetos, lo que para aquellos que generan inconsistencias le trata de modificar su clase, usando el grado de pertenencia aproximada (aunque esto no garantiza que el conjunto de aprendizaje resultante sea totalmente consistente). Edit2RS hace más flexible la definición de la aproximación inferior permitiendo cierto grado de inconsistencia (definido por el umbral que se establezca).

La función `edit2RS (list_conjuntos_aproximar, list_rasgos, alfa)` implementa el algoritmo pasándole además como atributo el umbral.

2.4.2 Evaluación de resultados

Para evaluar los resultados de aplicar los algoritmos de edición Edit1RS, Edit2RS y Edit3RS se utilizaron las bases de casos que se muestran en la tabla 3.

Base de casos	Cantidad de casos	Cantidad de atributos	Cantidad de clases
diabetes	768	9	2
heart-statlog	270	10	2
iris	150	4	3
glass	214	10	7
credit	690	15	2

Tabla 3. Información de las bases de casos utilizadas para evaluar la edición

Se realiza el análisis de los métodos edición a partir de comparar el número de ejemplos resultantes de aplicar los métodos a diferentes bases de datos. En la tabla 4 se muestra el efecto de reducción de los métodos, no aparece el método Edit3RS, pues este no elimina objetos sino que re-etiqueta los que pertenecen a las fronteras de las clases usando el grado de pertenencia de estos a las clases. Se puede apreciar una reducción significativa en la mayoría de los casos de la cantidad de objetos, mayor en el caso de Edit1RS.

Para analizar en qué medida esta reducción afecta la eficacia de la clasificación se hace un estudio usando varios métodos de clasificación conocidos.

Bases de casos	# de ejemplos sin editar	# de ejemplos Edit1RS	# de ejemplos Edit2RS
diabetes	768	176	499
heart_statlog	270	260	260
iris	150	117	147
glass	214	62	141
credit	690	623	655

Tabla 4. Resultados de evaluar métodos de reducción de ejemplos

La tabla 5 muestra los resultados de aplicar los clasificadores Naives Bayes (NV), Multilayer Percerceptron (MLP) y una versión mejorada de C4.5 (J48), implementadas todas en WEKA. Se utilizaron como conjuntos de prueba (TEST) la base de casos original (BC) con vista a tener en cuenta el valor máximo alcanzable, el 34% de la misma (34%BC), la

validación cruzada con 10 particiones(CV) y la base de casos editada por Edit2RS(BCedit) y como medida para evaluar el índice de Kappa. Considerando las columnas cuarta, quinta y sexta, se pueden apreciar resultados mejores con la base editada del índice de Kappa en la mayoría de los casos; los cuales aparecen en negrita.

	TEST	BC	34%BC	CV	BCedit2
BC	clasificador	Kappa	Kappa	Kappa	Kappa
diabetes	NB	0.416	0.45	0.402	0.40
	MLP	0.565	0.425	0.379	**0.55**
	J48	0.481	0.47	0.392	**0.49**
heart_stattlog	NB	0.707	0.63	0.663	**0.707**
	MLP	0.962	0.48	0.501	**0.947**
	J48	0.675	0.521	0.523	**0.675**
iris	NB	0.96	0.94	0.93	**0.96**
	MLP	0.97	0.94	0.93	**0.96**
	J48	0.96	0.94	0.96	**0.96**
glass	NB	0.552	0.49	0.494	**0.502**
	MLP	0.702	0.49	0.516	**0.608**
	J48	0.601	0.56	0.469	0.548
credit	NB	0.721	0.699	0.715	**0.727**
	MLP	0.924	0.802	0.689	**0.888**
	J48	0.762	0.691	0.702	**0.803**

Tabla 5. Resultados del índice de Kappa para diversos clasificadores

Cuando se hace el análisis similar de la eficacia usando los métodos Edit1RS y Edit3RS, los resultados muestran que la eficacia lograda con Edit1RS es menor y con Edit3RS mayor respecto a lo que alcanza Edit2RS.

2.5 Conclusiones parciales

Se implementó una biblioteca en Python que facilita el preprocesamiento de los datos utilizando métodos basados en conjuntos aproximados; la cual incluye un nuevo método de edición.

Se demuestra la eficacia de los métodos de reducción de datos incluidos en la biblioteca a partir de los experimentos desarrollados, demostrándose la posibilidad de utilizar esta para el análisis inteligente de datos.

Capítulo 3

Tratamiento de los conceptos de RST para grandes volúmenes de datos

3. TRATAMIENTO DE LOS CONCEPTOS DE LA RST PARA GRANDES VOLÚMENES DE DATOS

A menudo se requieren una gran cantidad de datos etiquetados, en los que la aproximación de conceptos y la reducción de atributos son dos cuestiones clave. Sin embargo, con el advenimiento de la era del Big Data, el etiquetado de datos es una tarea costosa y laboriosa y, a veces, incluso inviable, mientras que los datos no etiquetados son baratos y fáciles de recopilar. Por lo tanto, las técnicas para el análisis de datos en bruto en Big Data son deseables. En el presente capítulo se tratan los conceptos básicos de RST para grandes volúmenes de datos, en particular lo referido a clase de equivalencia

3.1 Incremento de eficiencia mediante vectorización

La vectorización es un tipo de procesamiento paralelo; el término se usa normalmente para referirse a la operación SIMD (instrucción única, datos múltiples). Permite que más hardware de la computadora se dedique a realizar el cálculo, por lo que el cálculo se realiza más rápido. Muchos problemas numéricos, especialmente la solución de ecuaciones diferenciales parciales, requieren que se realice el mismo cálculo para un gran número de células, elementos o nodos.

La vectorización utiliza hardware especial. A diferencia de una CPU multinúcleo, para la cual cada unidad de procesamiento paralelo es un núcleo de CPU completamente funcional, las unidades de procesamiento vectorial pueden realizar operaciones simples, y todas las unidades realizan la misma operación al mismo tiempo, operando en una secuencia de valores de datos (un vector) simultáneamente (McKinney, 2012) .

La vectorización tiene dos beneficios principales:

- ✓ El hardware diseñado para admitir instrucciones vectoriales generalmente tiene hardware que es capaz de realizar múltiples operaciones cuando se usan instrucciones vectoriales.
- ✓ Se ahorra gastos generales mediante el uso de instrucciones vectoriales.

3.1.1 Biblioteca Numpy en Python

En los últimos años, el soporte mejorado de la biblioteca de Python, ha convertido el lenguaje en una alternativa sólida para las tareas de manipulación de datos. Combinado con la solidez en la programación de propósito general, es una excelente opción como lenguaje único para crear aplicaciones centradas en datos.

El paquete Numpy (Numerical Python) es el paquete fundamental requerido para la computación científica de alto rendimiento y el análisis de datos y uno de los paquetes fundamentales para la manipulación de los datos. NumPy presenta varias funcionalidades para el análisis de datos; operaciones de matrices vectorizadas rápidas para la eliminación y limpieza de datos, subconjuntos y filtrado, transformación y cualquier otro tipo de cálculos. Estas operaciones favorecen la eficiencia computacional del código evitando el uso de bucles. Una de las características clave de NumPy es su objeto de matriz N-dimensional, o `ndarray`, que es un contenedor rápido y flexible para grandes conjuntos de datos en Python. Estas matrices le permiten realizar operaciones matemáticas en bloques enteros de datos usando una sintaxis similar a las operaciones equivalentes entre elementos escalares (Oliphant, 2006). Cada `ndarray` es una colección homogénea de exactamente el mismo tipo de datos, cada elemento ocupa el mismo tamaño de bloque de memoria. Estos arreglos son importantes porque permiten realizar operaciones iterativas sobre los datos sin necesidad de usar ciclos.

Para realizar operaciones en datos almacenados en `ndarray` se utiliza `ufunc` o función universal. Envolturas vectorizadas rápidas para funciones simples que toman uno o más valores escalares y producen uno o más resultados escalares, en (McKinney, 2012) se describen todas estas funciones. Algunas de estas funciones y operaciones entre matrices para eliminar el uso de ciclos y aumentar la eficiencia se aplicaron a los métodos de la biblioteca RoughPython.

3.1.2 Aplicación de técnicas de vectorización para la determinación de las clases de equivalencia y similitud

Para mejorar la eficiencia de los métodos implementados, en particular las clases de equivalencia y de similaridad, las cuales requieren aplicaciones muy costosas y el trabajo con grandes cantidades de datos se utilizó algunas técnicas y funciones de la biblioteca Numpy.

Primeramente, se creó el sistema de decisión como un `ndarray`, lo cual permite el acceso rápido a los datos y la aplicación de operaciones de forma más eficiente.

En el caso de las bases de casos discretas, se implementó el cálculo de las clases de equivalencia usando distancia Euclidiana, es decir, cada ejemplo tiene una distancia con respecto a los otros casos y si los mismos son similares la distancia entre ellos es 0, lo que significa que son equivalentes. Como las bases de casos son discretas es necesario enumerar los datos nominales, en la realización de esta operación también se aplican técnicas de vectorización. La distancia entre cada objeto se calcula mediante funciones del paquete Numpy como se ve en la figura 3.1.

```
def distancia_euclidiana(self, vector1, vector2):
    return np.sqrt(self.peso_rasgo_general * np.sum((np.array(vector1) - np.array(vector2)) ** 2))
```

Figura 3.1 Función distancia_euclidiana

La función `distancia_euclidiana(self,vector1,vector2)` calcula la distancia entre dos objetos de la base de casos, de ahí que se le pasen como parámetros a la función el vector1 y vector2 que son los valores de cada rasgo para cada uno de los objetos, es decir la fila correspondiente a cada instancia que se le desea calcular la distancia. Se utilizan la función sqrt(), sum() y otros operadores aritméticos de Numpy. El `peso_rasgo_general` es una variable interna que almacena el peso de cada rasgo. A cada clase de equivalencia pertenecen los objetos que tengan distancia 0 entre ellos.

También se implementa el cálculo de clases de similitud pero en este caso para bases de casos continuas, para crear la matriz se usaron técnicas de vectorización, esta matriz es un `ndarray` de nxn, donde n es la cantidad de objetos del universo. En esta matriz se tiene la distancia de cada ejemplo a los demás a partir de la función correspondiente (ver figura 3.1). A cada objeto se le calcula su clase de similitud, a la cual pertenecen aquellos objetos cuya distancia a él sea menor igual que un umbral (valor [0,1 introducido por el usuario). En este caso para el cálculo de la distancia entre objetos fue necesario normalizar los valores de cada uno de los rasgos porque los rasgos con dominios mayores inciden en los demás rasgos,

además se utiliza un peso para los rasgos, los cuales se dan por datos o se le puede asignar el valor 1/n a cada uno, cuando todos los rasgos tienen la misma importancia.

3.2 Implementación en la plataforma Spark de los conceptos básicos de la RST

Los métodos y definiciones de la RST y sus extensiones así como el cálculo de reductos y la selección de ejemplos requieren mucho tiempo de cómputo cuando se trabaja con muchos conjuntos de datos, especialmente Big Data. Con el objetivo de promover aplicaciones efectivas de la RST y sus extensiones en Big Data, se propone un diseño de los principales conceptos de esta teoría en el entorno Spark y la implementación de estos conceptos fundamentales en Spark, con el objetivo de obtener un rendimiento computacional eficiente.

Para abundar acerca de la implementación de la RST en Spark es necesario analizar su implementación en el enfoque MapReduce.

3.2.1 Big Data

El término, Big Data, ha sido creado para referirse a la gran cantidad de datos que no pueden ser manejados por los métodos o técnicas tradicionales de manejo de datos. El campo de Big Data desempeña un papel indispensable en varios campos, como agricultura, banca, educación, química, finanzas, computación en la nube, mercadeo, acciones de atención médica y la minería de datos en general.

El análisis de Big Data es el método para analizar grandes datos para revelar patrones ocultos, relaciones incomprensibles y otros datos importantes que se pueden utilizar para resolver decisiones mejoradas. Existe un interés cada vez mayor por Big Data debido a su rápido desarrollo y ya que cubre diferentes áreas de aplicaciones (Memon *et al.*, 2017).

Big data es un campo de avance. La velocidad y la variedad del desarrollo de datos crece debido a la expansión de sensores y teléfonos celulares con la asociación web. El uso de técnicas estadísticas convencionales no cumple los requisitos para analizar los datos (Wu *et al.*, 2013); de ahí la necesidad de desarrollar herramientas que permitan el procesamiento de grandes volúmenes de datos para poder extraer conocimiento de ellos.

El procesamiento y análisis de esta gran cantidad de datos representan una necesidad fundamental y un reto para extraer información útil ya que los algoritmos tradicionales de

Aprendizaje Automatizado (AA) y Minería de Datos (MD) no son capaces de escalar a este tipo de problemas (Chen and Zhang, 2014). Para enfrentar este reto han emergido soluciones de programación paralela para el manejo de datos a gran escala como el modelo MapReduce (Dean and Ghemawat, 2008) y Apache Spark (Zaharia *et al.*, 2012).

3.2.2 MapReduce

Existen diversas herramientas diseñadas por los gigantes tecnológicos. La aproximación más popular es el sistema de procesamiento distribuido MapReduce, presentado por Google en 2003.

El nuevo paradigma de programación para el procesamiento de datos masivos sigue la estrategia "divide y vencerás". Descompone el problema planteado, de gran tamaño, en un gran conjunto de problemas pequeños (bloques o fragmentos de datos) que se procesan en una cola virtual de tareas que se ejecutan conforme a las disponibilidades de ordenadores (función Map), y las salidas asociadas a los bloques de datos se fusionan para proporcionar una solución al problema (función Reduce). Esta función requiere diseñar algoritmos que integren los modelos o salidas del procesamiento de los bloques en un modelo o solución final para el problema (Herrera, 2014).

La función Map toma un par (llave, valor) para cada elemento en la entrada y genera como salida un conjunto de pares intermedios (llave, valor). Luego el marco de trabajo MapReduce recolecta todos los pares intermedios y los agrupa de acuerdo a la llave, generando un grupo para cada llave que constituye la entrada para la función Reduce. Esta última se aplica en paralelo a cada grupo producido por la función Map. Toma estos grupos y los combina para generar los correspondientes pares (llave, valor) como salida final del procedimiento MapReduce. De este modo, el programador se limita al desarrollo de las funciones Map y Reduce dejando todos los detalles de comunicación, balance de carga, asignación de recursos, inicio de trabajo y distribución de archivos a la plataforma subyacente. No todos los algoritmos pueden expresarse de acuerdo al esquema de MapReduce, por ejemplo, los algoritmos iterativos (Lin, 2013).

La teoría de los conjuntos aproximados resulta de interés para la minería de datos, la aproximación inferior y la aproximación superior son conceptos básicos de la RST, el cálculo efectivo de estas es vital para mejorar el rendimiento de la minería de datos, por lo que resulta

necesario realizar un método paralelo para calcular las aproximaciones. En (Zhang, Li and Pan, 2012) se propone un método paralelo para calcular las aproximaciones de la RST en el enfoque MapReduce.

3.2.3 Paralelización del cálculo de las aproximaciones inferiores y superiores de la RST en el enfoque MapReduce

El cálculo de las aproximaciones inferiores y superiores de la RST obtenidas por el método paralelo son las mismas que las obtenidas por el método en serie que se presenta en la figura 3.2. Pero al usar Map-Reduce, se ejecutan fases independientes en paralelo como la clase de equivalencia computacional, la clase de decisión computacional, la construcción de asociaciones basadas en Map-Reduce. Por lo tanto, el tiempo requerido es muy inferior en comparación con el método tradicional de cálculo de conjuntos aproximados (Zhang, Li and Pan, 2012).

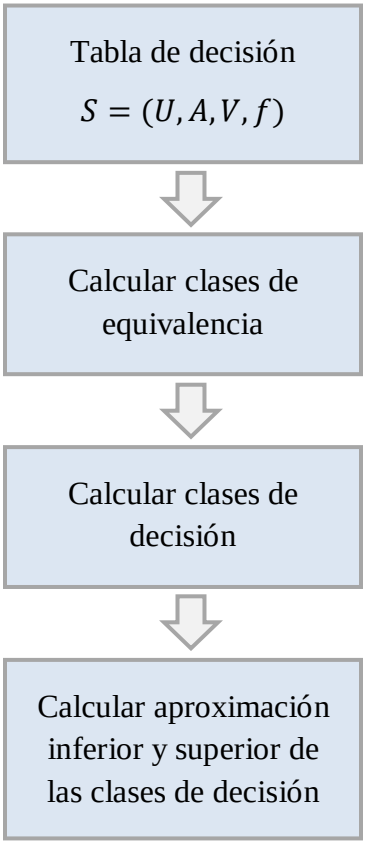

Figura 3.2 Método en serie para el cálculo de las aproximaciones

Clases de equivalencia y clases de decisión en paralelo

Para realizar el cálculo de las aproximaciones en paralelo, se obtienen primeramente las clases de equivalencia en paralelo con MapReduce a partir de los algoritmos PACRSEC-Map y PACRSEC-Reduce (Zhang *et al.*, 2012) .

En la figura 3.4 se muestra el resultado de aplicarle los algoritmos PACRSEC a un sistema de decisión S (ver figura 3.3), que se subdivide en dos $S_1 = (U_1, A = \{C \cup D\}, V_a, f)$ y $S_2 = (U_2, A = \{C \cup D\}, V_a, f)$ donde $U = U_1 \cup U_2$ es el universo, C los rasgos por condición, D los rasgos decisión, V_a el dominio de los atributos y f una función de información tal que $f: U \times A \Rightarrow f(x, A) \in V_a$.

Table 2
A decision table S_1.

Object	a_1	a_2	D
x_1	0	0	0
x_2	0	0	1
x_3	1	0	1
x_4	1	1	1
x_5	0	0	0
x_6	1	1	2

Table 3
A decision table S_2.

Object	a_1	a_2	D
x_7	0	0	0
x_8	1	1	1
x_9	1	1	2
x_{10}	0	1	1
x_{11}	1	0	2
x_{12}	1	1	1

Figura 3.3 Ejemplo de sistemas de decisión

Las clases de equivalencia que se obtienen a partir del subconjunto de rasgos $B = \{a_1 y\ a_2\}$ es $U/B = \{E_1, E_2, E_3, E_4\}$.

Al aplicar PACRSEC-Map se obtienen las clases de equivalencia para cada una de las particiones de U y al aplicar PACRSEC-Reduce se combinan si su conjunto de información es el mismo. Las clases de decisión para un sistema de decisión se pueden ejecutar de igual forma en paralelo a partir de los algoritmos PACRSDC-Map y PACRSDC-Reduce (Zhang

et al., 2012). Los algoritmos PACAED-Map y PACAED-Reduce permiten la asociación entre las clases de equivalencia y las clases decisión.

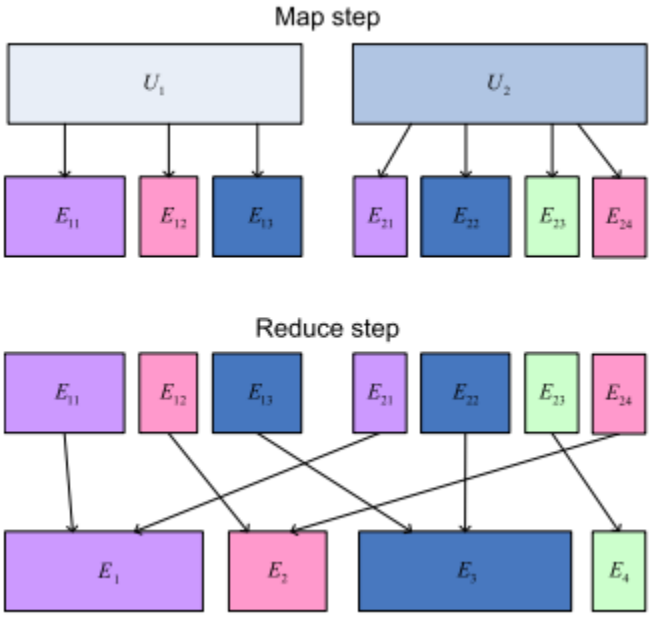

Figura 3.4 Clases de equivalencia de S usando MapReduce

Aproximación inferior y Aproximación superior

Las aproximaciones inferiores y superiores se calculan mediante asociaciones entre las clases de equivalencia y las clases decisión. Si se calculan las aproximaciones directamente, la memoria puede desbordarse por la cantidad de objetos que tienen los conjuntos de datos, por lo que se diseña el algoritmo ACIRSAA (Algoritmo para el cálculo de los índices de aproximaciones de los conjuntos aproximados por asociaciones), en (Zhang *et al.*, 2012) (Zhang, Li and Pan, 2012).

Después de aplicar ACIRSAA se generan las aproximaciones a partir de estos. La figura 3.5 muestra el diagrama para el cálculo de las aproximaciones de forma paralela.

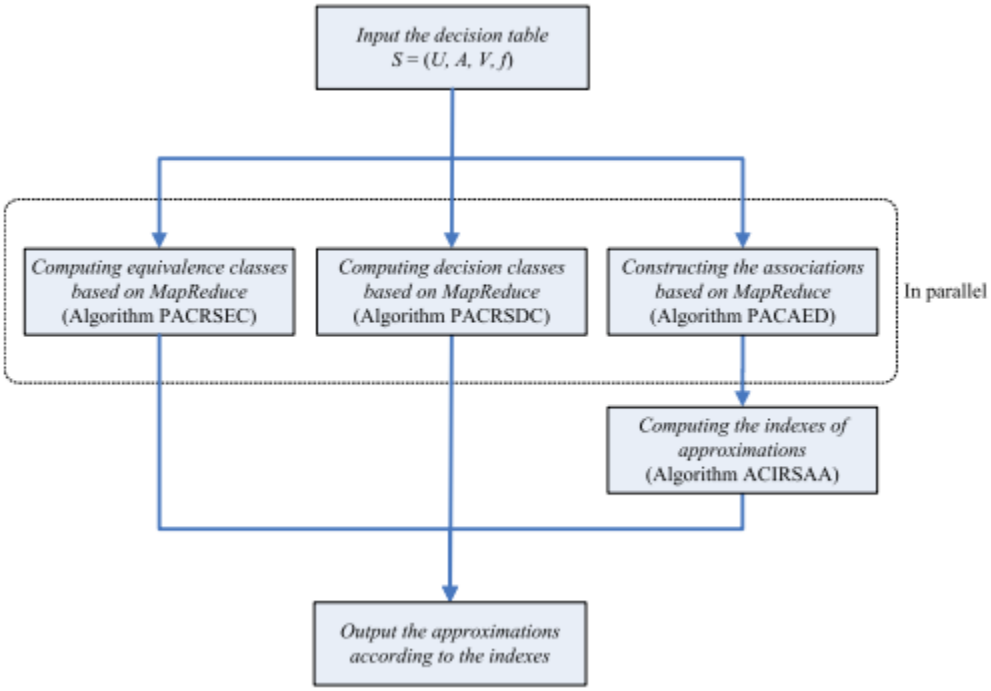

Figura 3.5 Diagrama del cálculo de aproximaciones en paralelo

3.2.4 Apache Spark

Apache Spark es un sistema para el procesamiento de datos masivos, centrado en la velocidad por el procesamiento en memoria, facilidad de uso y un análisis sofisticado para el diseño de algoritmos. Aborda muchas de las limitaciones de MapReduce desde su propia concepción.

Spark extiende el modelo MapReduce y soporta eficientemente más tipos de trabajos computacionales, incluyendo consultas interactivas y procesamiento de flujo de datos. La velocidad es importante cuando se trata de procesar grandes conjuntos de datos y es una de las principales características que Spark ofrece, para esto hace un uso intensivo de la memoria computacional (Karau et al. ,2015).

La principal abstracción de programación en Spark son los llamados Resilient Distributed Dataset (RDD), colección de objetos inmutables y distribuidos. Cada RDD es particionado en varias nuevas particiones, las cuales son procesadas en diferentes nodos del clúster. RDD pueden contener varios tipos de objetos, que pueden ser de distintos lenguajes, ya sea Java, Scala, o Python, además de clases definidas por el usuario. En Spark el procesamiento se expresa creando nuevos RDD, transformando RDD existentes, o llamando operaciones sobre un RDD para obtener un resultado. Oculto a la vista del usuario, Spark automáticamente

distribuye los datos contenidos en los RDD en el clúster y paraleliza las operaciones que realiza sobre los mismos (Karau et al., 2015).

Se pueden crear RDD de dos formas, cargando datos externos o distribuyendo una colección de objetos en su programa controlador. Una vez creados, los RDD ofrecen dos tipos de operaciones sobre ellos, transformaciones y acciones, las primeras construyen un nuevo RDD a partir de uno ya existente, y las segundas computan resultados a basados en un RDD, y estos son devueltos al programa controlador o salvados en un sistema de archivos externo.

Entre las transformaciones más usadas sobre RDD están; Map; que aplica una función a cada elemento del RDD y retorna un nuevo RDD con el resultado y Filter; que retorna un RDD que contiene los elementos que pasaron la condición enviada para filtrar.

Las acciones básicas sobre un RDD son collect(); retorna todos los elementos de un RDD, count(); retorna el número de elementos en un RDD, take(num); retorna el número de elementos del RDD, reduce(func); combina los elementos del RDD juntos en paralelo.

Una de los atributos más importantes de Spark es que puede "cachear" datos en memoria por todo el clúster, es decir, esta es una de las más importantes utilidades de los RDD. Spark retiene los RDDs en memoria, así la primera vez que éstos se utilizan en una acción, son cargados desde la fuente de los datos, y cuando sea necesario la reutilización de los mismos, los datos se acceden directamente desde la memoria, evitándose así las altas penalizaciones de entrada/salida. Esta propiedad es muy importante cuando se realiza un trabajo sobre grandes conjuntos de datos, o se utilizan algoritmos de Aprendizaje Automatizado que requieren varias pasadas sobre los mismos datos (Pentreath, 2015).

Apache Spark incluye otros componentes de primer orden como Spark SQL, Spark MLlib, Spark ML, Spark Streaming y GraphX los cuales proveen funcionalidades de procesamiento más específicas.

3.2.5 Implementación en el framework de Apache Spark

La implementación de RST en Apache Spark puede beneficiarse de las optimizaciones propias de este framework.

Spark SQL es una de las componentes de esta plataforma, permite a los programadores de Spark aprovechar los beneficios del procesamiento relacional (por ejemplo, consultas

declarativas y almacenamiento optimizado), y permite a los usuarios de SQL llamar a bibliotecas de análisis complejo en Spark. Además incluye un optimizador altamente extensible, Catalyst, creado con las características del lenguaje de programación Scala, que facilita la generación de códigos de control y la definición de puntos de extensión (Armbrust et al., 2015).

Spark SQL introduce dos tipos de datos semi-estructurados en forma de interfaces, DataFrames y Datasets. Este componente constituye el futuro de Spark, con opciones de almacenamiento más eficientes, optimizadores avanzados y operaciones directas en datos serializados. Este componente es fundamental para extraer el mejor desempeño de Spark (Karau and Warren, 2017). Por estas razones y para la implementación de la RST en Apache Spark se utilizó Spark SQL.

Para visualizar como se realiza el procedimiento se utilizó una base de casos pequeña de dos rasgos predictores y un rasgo decisión con 12 objetos. La tabla 6 muestra los objetos representados por los rasgos predictores.

Al aplicar la consulta siguiente:

```
tabla6.select(col("features").as("feat"),col("object")).groupBy(co
l("feat")).agg(collect_set(col("object")).alias("objects"))
```

Se obtienen las clases de equivalencia en la columna `object` de la Tabla 7:

object	features
1	[no, no]
2	[no, high]
3	[yes, no]
4	[yes, very high]
5	[no, no]
6	[yes, high]
7	[no, no]
8	[no, high]
9	[yes, high]
10	[yes, very high]
11	[no, no]
12	[yes, high]

Tabla 6

feat	objects
[no, no]	[1, 5, 7, 11]
[yes, no]	[3]
[yes, high]	[12, 9, 6]
[yes, very high]	[10, 4]
[no, high]	[2, 8]

Tabla 7

features	objects	class	feat
[yes, high, yes]	[12, 9]	yes	[yes, higth]
[no, no, no]	[1, 5, 7, 11]	no	[no, no]
[yes, high, no]	[6]	no	[yes, higth]
[yes, very high, yes]	[10, 4]	yes	[yes, very higth]
[no, high,yes]	[2, 8]	yes	[no, higth]
[yes, no, yes]	[3]	yes	[yes, no]

Tabla 8

En la tabla 8 se obtienen los objetos que tienen iguales rasgos predictores e igual clase de decisión.

Al aplicar `join` entre las tablas 7 y 8 por la columna `feat` se obtiene la Tabla. 9, en la cual se obtienen los objetos que se repiten para diferentes clases, es decir las clases de equivalencia que tengan objetos con diferente decisión. Para cada clase de decisión se obtienen las aproximaciones superiores representadas en la Tabla.10 que son el resultado de aplicar la consulta siguiente:

```
tabla7.join(Tabla8,usingColumn =
"feat").select(Tabla7.col("objects"),col("class")).groupBy(col("cl
ass")).agg(collect_list(col("objects")))
```

feat	objects	features	objects2	class
[yes, higth]	[12,9,6]	[yes, high, yes]	[12, 9]	yes
[no, no]	[1, 5, 7, 11]	[no, no, no]	[1, 5, 7, 11]	no
[yes, higth]	[12,9,6]	[yes, high, no]	[6]	no
[yes, very higth]	[10, 4]	[yes, very high, no]	[10, 4]	yes
[no, higth]	[2, 8]	[no, high,yes]	[2, 8]	yes
[yes, no]	[3]	[yes, no, yes]	[3]	yes

Tabla 9

Clases	Aproximación Inferior	Aproximación Superior
yes	[2,8,3,10,4]	[2,8,3,4,10,6,9,12]
no	[1,5,7,11]	[1,5,7,11,6,9,12]

Tabla 10

El cálculo de las clases de equivalencia y aproximaciones superiores se realiza de forma eficiente. Para generar las aproximaciones inferiores se utiliza la operación de intersección que es computacionalmente más costosa por lo que es necesario más tiempo de ejecución y disponibilidad de memoria.

El diseño e implementación de los conceptos básicos de RST en Spark hace factible la aplicación de esta teoría para grandes volúmenes de datos. Los tiempos de ejecución para conjuntos de datos voluminosos generados en el experimento desarrollado así lo demuestran, donde se reducen sustancialmente los tiempos para grandes volúmenes de datos.

La Tabla.11 muestra los resultados de los tiempos de ejecución en minutos para calcular las aproximaciones inferior y superior para dos conjuntos de datos. Nótese que un incremento sustancial de la cantidad de objetos no produce un incremento significativo del tiempo de ejecución, a pesar de que de acuerdo a los estudios presentados en (Bell and Guan, 1998) y (Deogun *et al.*, 1998), la complejidad computacional de encontrar la aproximación inferior y la superior es O (lm^2), donde l es el número de atributos que describen los objetos y m es la cantidad de objetos en el universo.

Nombre	Cantidad de Instancias	Cantidad de atributos	Tiempo de Ejecución
data1	50000	10	1 min 40 sec
data2	1000000	50	3 min 9 sec

Tabla 11

3.3 Conclusiones parciales

La implementación de la vectorización en la biblioteca PythonRST con el uso de NumPy y el ndarray reduce el costo computacional que engendra la implementación de los conceptos

básicos de RST y por ende aumenta la eficiencia computacional de los algoritmos de selección de rasgos o edición de conjuntos de entrenamiento que usan estos.

El diseño e implementación de los conceptos básicos de RST en Spark hace factible la aplicación de esta teoría para grandes volúmenes de datos, destacándose el hecho de que el incremento del costo computacional del cálculo de las aproximaciones no es proporcional al incremento de la cantidad de instancias.

4. CONCLUSIONES

- ✓ Se implementó una biblioteca en Python para el preprocesamiento de datos basado en RST y sus extensiones la cual ha sido evaluada satisfactoriamente usando bases de casos del *UCI Repository* atendiendo a la eficacia de los clasificadores.

- ✓ Se aplicó la vectorización a distintos métodos de la biblioteca fundamentalmente al cálculo de las clases de equivalencia y de similitud mejorándose la eficiencia de estos.

- ✓ Se implementaron los principales conceptos de RST en el entorno Spark para Big Data; en los experimentos realizados se evidencia que se reduce significativamente el costo en tiempo computacional para el cálculo de las aproximaciones cuando crece significativamente la cantidad de objetos.

5. RECOMENDACIONES

- ✓ Incluir otros métodos de preprocesamiento de datos en la biblioteca.
- ✓ Ampliar la vectorización en otros métodos de la biblioteca.
- ✓ Continuar la implementación de las restantes definiciones y métodos de la RST en el entorno Spark.
- ✓ Optimizar el cálculo de las aproximaciones inferiores.

6. REFERENCIAS BIBLIOGRÁFICAS

1. Aha, D. W. (1997) *Lazy learning*. Kluwer Academic Publishers. Available at: https://books.google.com.cu/books.
2. Ahn, B. S. (2000). *The integrated methodology of rough set theory and artificial neural networks for business failure predictions*
3. Arco, L., R. Bello, et al. (2006). *On clustering validity measures and the Rough Set Theory*. 5th Mexican International Conference on Artificial Intelligence, IEEE Computer Society Press
4. Armbrust, M. *et al.* (2015) *'Spark SQL: Relational Data Processing in Spark'*, in Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data - SIGMOD '15. New York, New York, USA: ACM Press, pp. 1383–1394.
5. Bache, K., & Lichman, M. (2013). UCI machine learning repository.
6. Bazan, J., N. H. Son, et al. (2003). A View on Rough Set Concept Approximations. Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing. 9th International Conference, RSFDGRC2003, Chongqing, China.
7. Bello, R., A. Nowe, et al. (2006). *"Two Step Ant Colony System to Solve the Feature Selection Problem"*, Lectures Notes on Computer Sciences. Springer-Verlag: 588- 596.
8. Bell, D. A. and Guan, J. W. (1998) *'Computational methods for rough classification and discovery'*, Journal of the American Society for Information Science. Wiley Online Library, 49(5), pp. 403–414.
9. Bello, R., Garcíia, M. and Pérez, J. N. (2012) *'Teoría de los conjuntos aproximados: conceptos y métodos computacionales'*, Universidad Distrital Francisco José de Caldas. Bogotá, Colombia.
10. Bhargava, N. *et al.* (2013) *'Decision Tree Analysis on J48 Algorithm for Data Mining'*, Intrenational Journal of Advanced Research in Computer Science and Software Engineering, 3(6).

11. Borgelt, C. and R. Kruse (2003). Neural Networks, Working Group Neural Networks and Fuzzy Systems. Otto-von-Guericke. University of Magdeburg.

12. Bueno, E. *Estado del arte y tendencias en creación y gestión del conocimiento*. in Congreso Iberoamericano de Gestión del Conocimiento y la Tecnología (IBERGECYT 2001). 2001. La Habana, Cuba.

13. Caballero, Y. (2005). Uso de los Conjuntos Aproximados para el tratamiento de los datos. Santa Clara, Cuba, Universidad Central de Las Villas.

14. Caballero, Y. (2007) *'Aplicación de la Teoría de los Conjuntos Aproximados en el Preprocesamiento de los Conjuntos de Entrenamiento para Algoritmos de Aprendizaje Automatizado'*, Universidad Central Marta Abreu de Las Villas, Santa Clara, Cuba.

15. Caballero, Y., R. Bello, et al. (2006)b. *Improving the k-NN method: Rough Set in edit training set*. The First IFIP International Conference on Artificial Intelligence in Theory and Practice, Springer Boston.

16. Caballero, Y., R. Bello, et al. (2007). La Teoría de los Conjuntos Aproximados en el mejoramiento de los conjuntos de entrenamiento en Bioinformática. II Congreso Internacional de Bioinformática y Neuroinformática. Informática 2007, La Habana, Cuba.

17. Canals, A., M. Boisot, and A. Cornella, *Gestión del conocimiento*. 2003, Gestión: 2000: Barcelona, España.

18. CHEN, C. L. P. and ZHANG, C. Y. (2014) *Data-intensive applications, challenges, techniques and technologies:* A survey on Big Data. Information Sciences, 275, 314–347.

19. Chin, K. S., J. Liang, et al. (2003). Rough Set Data Analysis Algorithms for Incomplete Information Systems. Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing. 9th International Conference, RSFDGRC2003, Chongqing, China.

20. Cover, T. M. and P. E. Hart (1967). "Nearest neighbour pattern classification." Institute of Electronical and Electronics Engineers Transactions on Information Theory 13: 21-27.
21. Dean, J. and Ghemawat, S. (2008) *'MapReduce: simplified data processing on large clusters'*, Communications of the ACM. ACM, 51(1), p. 107.
22. Dean, J. and Ghemawat, S. (2008) *'MapReduce: simplified data processing on large clusters'*, Communications of the ACM. ACM, 51(1), p. 107. doi: 10.1145/1327452.1327492.
23. Deogun, J. S. (1995). Exploiting upper approximations in the rough set methodology. First International Conference on Knowledge Discovery and Data Mining, Canada.
24. Deogun, J. S. et al. (1998) 'Feature selection and effective classifiers', Journal of the American Society for Information Science. Wiley Online Library, 49(5), pp. 423–434.
25. Dorigo, M., Birattari, M., Stützle, T. (2010). Ant colony optimization, in Encyclopedia of Machine Learning, Springer: New York City. p. 36-39.
26. Dunstsh, I. & G. Gunter. (2000). "Rough set data analysis."
27. Fayyad, U., Piatetsky-Shapiro, G. and Smyth, P. (1996) *'The KDD process for extracting useful knowledge from volumes of data'*, Communications of the ACM. ACM, 39(11), pp. 27–34.
28. Fernández, A. *et al.* (2014) *'Big Data with Cloud Computing: an insight on the computing environment, MapReduce, and programming frameworks'*, *WIREs Data Mining Knowl Discov*, 4, pp. 380–409. doi: 10.1002/widm.1134.
29. Fernando, J., Saldaña, R. and Flores, R. G. (2005) *El proceso de descubrimiento de conocimiento en bases de datos*.
30. Filiberto, Y. *et al.* (2011) Revista Facultad de Ingeniería, *Una medida de la teoría de los conjuntos aproximados para sistemas de decisión con rasgos de dominio continuo*. Universidad de Antioquia.
31. Frappaolo, C., *Knowledge Management*. 2006, West Sussex, England: Capstone Publishing Ltd. (A Wiley company).

32. García, A. (2012) *Inteligencia artificial : fundamentos, práctica y aplicaciones*. RC Libros.
33. García, M., Y. Villuendas, et al. (2005). Métodos de selección y construcción de objetos para el mejoramiento de un clasificador supervisado: estado del arte.
34. García, S. *et al.* (2016) '*Big Data : Preprocesamiento y calidad de los datos'*, Novática.
35. García, S., Luengo, J., & Herrera, F. (2015). *Data preprocessing in data mining*. Springer.
36. García, S., Luengo, J., & Herrera, F. (2016). *Data preprocessing in data mining*. Springer.
37. Goldberg, D.E. (1989). Genetic algorithms in optimization, search and machine learning, Boston, USA: Addison-Wesley Longman Publishing Co., Inc.
38. Grabowski, A. (2003). "Basic Properties of Rough Sets and Rough Membership Function." Journal of Formalized Mathematics 15.
39. Greco, S. (2001). "Rough sets theory for multicriteria decision analysis." European Journal of Operational Research 129: 1-47.
40. Grzymala-Busse, J. W. and S. Siddhaye (2004). Rough set approaches to rule induction from incomplete data. 10th International Conference on Information Processing and Management of Uncertainty in Knowledge-Bases systems IPMU2004, Perugia, Italy.
41. Hernández,C. & Rodriguez Rodríguez, J. E. (2013) *Preprocesamiento de datos esctructurados, Revista vínculos, 4(2)*.
42. Herrera, F. (2014). Big data: Procesando los datos en la sociedad digital. Revista Española de Física, 28(4), 40–44.
43. Herrera, F. and Cano, J. R. (2006) *'Técnicas de reducción de datos en KDD. El uso de Algoritmos Evolutivos para la Selección de Instancias'*, Actas del I Seminario sobre Sistemas Inteligentes (SSI_06), pp. 165–181.
44. Karau, H. and Warren, R. (2017) *High performance Spark: best practices for scaling and optimizing Apache Spark*. ' O'Reilly Media, Inc.'
45. Karau, H. et al.(2015*)* *Learning Spark* M. B. Ann Spencer, ed., O'Reilly Media,

Inc.

46. Kennedy, J. (2011). Particle swarm optimization, in Encyclopedia of Machine Learning, Springer: Boston, MA. p. 760-766.

47. Komorowski, J. and Z. Pawlak (1999). "Rough Sets: A tutorial." Rough Fuzzy Hybridization: A new trend in decision-making. Springer: 3-98.

48. Lin, J. (2013) *MapReduce is Good Enough? If All You Have is a Hammer*, Throw Away Everything Thats Not a Nail! Big Data, 1, 28-37.
49. Malvicino, F. and Yoguel, G. (2016) *Big data : avances recientes a nivel internacional y perspectivas para el desarrollo local*. Available at: www.ciecti.org.ar (Accessed: 23 May 2019).
50. Mayer-Schönberger, V. and Cukier, K. (2013) *'Big data: la revolución de los datos masivos'.*
51. McKinney, W. (2012) *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython*.
52. Memon, M. A. *et al.* (2017) 'Big Data Analytics and Its Applications', *Annals of Emerging Technologies in Computing*, 1(1), pp. 45–54. doi: 10.33166/AETiC.2017.01.006.
53. Mitchell, T. M. (1997). *Machine Learning*. *In Elements*. https://doi.org/10.1007/978-0-387-84858-7
54. Oliphant, T. E. (2006) *Guide to NumPy*. Available at: http://www.trelgol.com (Accessed: 29 May 2019).
55. Pal, S. K. and A. Skowron (1999). Rough Fuzzy Hybridization: A New Trend in DecisionMaking.

56. Parsons, S. (1996). "Current approaches to handling imperfect information in data and knowledges bases." IEEE Trans. On knowledge and data enginnering 8(3).

57. Pawlak, Z. (1982). "Rough Sets." International journal of Computer and Information Sciences 11: 341-356.

58. Pawlak, Z. (1982). "Rough Sets." International journal of Computer and Information Sciences 11: 341-356.

59. Pawlak, Z., S.K.M. Wong, W. Ziarko. (1988). *Rough sets: probabilistic versus deterministic approach*, International Journal of Man–Machine Studies 29: 81–95.

60. Pawlak, Z., S.K.M. Wong, W. Ziarko. (1988). Rough sets: probabilistic versus deterministic approach, International Journal of Man–Machine Studies 29: 81–95.
61. Pentreath, N., 2015. *Real-time Machine Learning with Spark Streaming* (Book Section)
62. Peters, J. F. (2005). Rough Ethology: Towards a Biologically-Inspired Study of Collective Behavior in Intelligent Systems with Approximation Spaces. Transactions on Rough Sets III. Lecture Notes in Computer Science. J. F. Peters and A. Skowron, Springer-Verlag GmbH. 3400.

63. Qian, Y. *et al.* (2018) 'Local rough set: A solution to rough data analysis in big data', *International Journal of Approximate Reasoning*. Elsevier, 97, pp. 38–63. doi: 10.1016/J.IJAR.2018.01.008.
64. Quinlan, J. R. (1993). *C-4.5: Programs for machine learning*. San Mateo, California.

65. Ruiz, R. (2006). *Heurísticas de selección de atributos para datos de gran dimensionalidad*. Departamento de Lenguajes y Sistemas Informáicos. Sevilla, Universidad de Sevilla.
66. Russell, S. J., & Norvig, P. (2016). *Artificial Intelligence : A Modern Approach*. Retrieved from http://thuvien.thanglong.edu.vn:8081/dspace/handle/DHTL_123456789/4010
67. Russom, P. and Org, T. (2011). *Big Data Analytics*. Available at: https://vivomente.com/wp-content/uploads/2016/04/big-data-analytics-white-paper.pdf
68. Salloum, S. *et al.* (2016) *'Big data analytics on Apache Spark'*, International Journal of Data Science and Analytics. Springer International Publishing, 1(3–4), pp. 145–164. doi: 10.1007/s41060-016-0027-9.

69. Simon, H. A. (1983). *Why should machines learn? Machine Learning, An Artificial Intelligence approach*. R. S. Michalski, J. G. Carbonell and T. M. Mitchel. Palo Alto, CA.

70. Skowron, A. (1999). *New directions in Rough Sets, Data Mining, and Granular Soft Computing*. 7th International Workshop (RSFDGRC'99), Yamaguchi, Japan, Lecture Notes in Artificial Intelligence 1711.

71. Skowron, A. and J. F. Peters (2003). Rough Sets: Trends and Challenges. Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing 9th International Conference, RSFDGRC 2003, Chongqing, China, LNCS 26392003.

72. Skowron, A. and J. Stepaniuk (1992). Intelligent systems based on rough set approach. International Workshop Rough Sets. State of the Art and Perspectives.

73. Slowinski, R. and D. Vanderpooten (1997). Similarity relation as a basis for rough approximations. Advances in Machine Intelligence & Soft-Computing. IV: 17-33.

74. Slowinski, R. and D. Vanderpooten (2000). "A generalized definition of rough approximations based on similarity." EEE Transactions on Data and Knowledge Engineering **12**(2): 331-336.

75. Sucar, L. E. (2006) 'Redes bayesianas', *BS Araujo, Aprendizaje Automático: conceptos básicos y avanzados*, pp. 77–100.

76. Verdenius, F. and van Someren, M. W. (1997) *'Applications of inductive learning techniques: a survey in the Netherlands'*, AI communications. IOS Press, 10(1), pp. 3–20.

77. Wang, X., Yang, J., Jensen, R., & Liu, X. (2006). *Rough set feature selection and rule induction for prediction of malignancy degree in brain glioma*. Computer Methods and Programs in Biomedicine, 83, 147–156.

78. Wang, X., Yang, J., Teng, X., Xia, W., & Jensen, R. (2007). Feature selection based on rough sets and particle swarm optimization. Pattern Recognition Letters, 28, 459–471.

79. Wilson, D. R. and T. R. Martínez (1997). "Improved Heterogeneous Distance Functions." Journal of Artificial Intelligence Research 6: 1-34.

80. Witten, I, Frank, E., Hall M., Pal, C. (2017). *Data Mining: Practical Machine Learning Tools and Techniques*, Fourth Edition. ELSEVIER, ISBN:978-0-12-804291-5
81. Witten, I. and E. Frank (2005). Implementations: Real machine learning schemes. Data Mining. Practical Machine Learning Tools and Techniques. I. Witten and E. Frank, Department of Computer Science. University of Waikato: 187-283.
82. Wróblewski, J. (1995). Finding Minimal Reducts using Genetic Algorithm. Proc. of the Second Annual Join Conference on Information Sciences. Wrightsville Beach, NC. Also in: ICS Research report 16/95, Warsaw University of Technology.
83. Wróblewski, J. (1998) 'Genetic Algorithms in Decomposition and Classification Problems', in. Physica, Heidelberg, pp. 471–487. doi: 10.1007/978-3-7908-1883-3_24.
84. Wu, X. *et al.* (2013) 'Data mining with big data', *IEEE transactions on knowledge and data engineering*. IEEE, 26(1), pp. 97–107.
85. Yang, J. and Honavar, V. (1998) 'Feature Subset Selection Using a Genetic Algorithm', in *Feature Extraction, Construction and Selection*. Boston, MA: Springer US, pp. 117–136. doi: 10.1007/978-1-4615-5725-8_8.
86. Zaharia, M. *et al.* (2012) *'Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing'*, in Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation, p. 2.
87. Zhang, J. *et al.* (2012) *'A parallel method for computing rough set approximations'*, Information Sciences. doi: 10.1016/j.ins.2011.12.036.
88. Zhang, J., Li, T. and Pan, Y. (2012) *'Parallel rough set based knowledge acquisition using MapReduce from big data'.*
89. Zhong, N., J. Dong, et al. (2001). "Using Rough sets with heuristics for feature selection." Journal of Intelligent Information Systems 16: 199-214.