Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior (Jaén)

Trabajo Fin de Grado

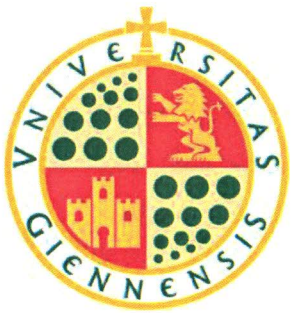
DESARROLLO DE UNA LIBRERÍA DE ALGORITMOS DE EXTRACCIÓN DE REGLAS DESCRIPTIVAS EN R Y DE LA INTERFAZ DE USUARIO ASOCIADA.

Alumno: Ángel Miguel García Vico

Tutora: Prof. Dña. María José del Jesús Díaz
Dpto: Departamento de Informática

Tutor: D. Francisco Charte Ojeda
DNI: 25996209
Universidad: Universidad de Granada

Junio, 2015

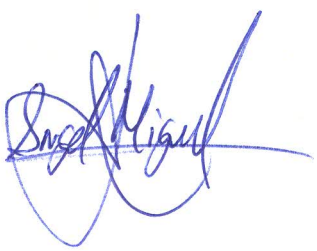


Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Doña MARIA JOSÉ DEL JESUS DÍAZ y Don FRANCISCO CHARTE OJEDA, tutores del Proyecto Fin de Carrera titulado: DESARROLLO DE UNA LIBRERÍA DE ALGORITMOS DE EXTRACCIÓN DE REGLAS DESCRIPTIVAS EN R Y DE LA INTERFAZ DE USUARIO ASOCIADA, que presenta ÁNGEL MIGUEL GARCÍA VICO, autorizan su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, JUNIO de 2015

El alumno:



ÁNGEL MIGUEL
GARCÍA VICO

Los tutores:

MARÍA JOSÉ
DEL JESÚS DÍAZ

FRANCISCO
CHARTE OJEDA

Agradecimientos

Con este proyecto se cierra una etapa importante de mi vida, pero se abre otra quizá mas importante aún. Hay tantísimas personas a las que quiero dedicarles unas palabras que podría escribir otra memoria y, a pesar de que no estéis nombrados explícitamente, estoy seguro de que sabéis lo agradecido que estoy.

- A mis padres, por haber tenido siempre esa confianza ciega y por su apoyo incondicional en todo momento. Aquí tenéis el fruto de todo vuestro esfuerzo.
- A mi hermano, porque sin él nada hubiera sido lo mismo. Sergio, eres un imbécil, pero aún así te admiro profundamente. Nunca cambies.
- A María, gracias por ese apoyo incondicional, por esa confianza plena, por entenderme y por quereme tanto. Gracias, gracias de todo corazón. Esta memoria te la dedico.
- A todos mis amigos, por todos esos momentos de risas y por lo buenos ratos que pasamos cuando nos juntamos.
- A mis compañeros del seminario 154, por hacerme las mañanas más amenas.
- A Pedro González, por tener que aguantarme sin necesidad.
- Y por último a mis tutores de proyecto. A María José por darme la oportunidad de realizar este proyecto y a Paco Charte por su ayuda y apoyo en los momentos más críticos de este proyecto.

A todos vosotros.

Muchas gracias.

Índice

1. Introducción, definición y planificación del Proyecto 12
1.1. Introducción 12
1.2. Motivación 14
1.3. Definición y Planificación del Proyecto 16
Definición de los objetivos 16
Estructura de la memoria del Proyecto 16
1.4. Agenda y planificación del proyecto 17
1.3.1. Planificación del Proyecto 17
1.3.2. Estimación de costes 19
2. Introducción a la minería de datos 24
1.4.1.
1.4.2. 2.1. Minería de datos 24
2.2. Descubrimiento de subgrupos 25
2.3. Medidas de calidad para el descubrimiento de subgrupos 27
2.4. Principales algoritmos de descubrimiento de subgrupos 29
Extensiones de algoritmos de clasificación. 29
2.4.1. Algoritmos basados en clasificación 29
2.4.2.
2.4.3. Extensión de algoritmos de reglas de asociación 30
2.4.4. Algoritmos evolutivos 30
2.5. Lógica Difusa 31
2.6. Estudio de librerías y plataformas existentes para el descubrimiento de subgrupos 35
3. Algoritmos de extracción de reglas de descubrimiento de subgrupos 37
3.1.1.
3.1.2. 3.1. Algoritmo SDIGA (Subgroup Discovery Iterative Genetic Algorithm) 37
Funcionamiento del algoritmo genético de SDIGA 38
Esquema de representación 38
3.1.2.1. Reglas canónicas 39
3.1.2.2. Reglas DNF 40
3.1.2.3. Operadores genéticos 40
3.1.3. 3.1.2.4. Operador de cruce 41
3.1.2.5. Operador de mutación 41
3.2.1. 3.1.2.6. Función de evaluación 41
3.1.2.7. Operador de reemplazo 43
Optimización Local 43
3.2. Algoritmo MESDIF (Multiobjetive Evolutionary Subgroup Discovery Fuzzy rules) ..44
Funcionamiento del algoritmo genético de MESDIF 44

Esquema de representación....45
Generación de la población inicial....45
Función de evaluación. 45
Función de truncado 46
Función de rellenado....47
Operadores genéticos....47
3.3. Algoritmo NMEEF-SD 48
Operadores genéticos....49
Operador de reinicialización....49
Operador de selección. Crowding distance. 50
4. Análisis, diseño e implementación de la librería de algoritmos. 52
4.1. Análisis de requerimientos 52
Requerimientos funcionales. 52
Requerimientos no funcionales 53
Casos de uso....53
4.2. Diseño....56
Diseño de clases del Sistema 56
4.2.1.1. Diagrama de clases de SDIGA....56
4.2.1.2. Diagrama de clases de MESDIF 58
4.2.1.3. Diagrama de clases de NMEEF-SD 59
Diseño de la secuencia de actividades del Sistema 60
4.2.2.1. Diagrama de secuencia de SDIGA....60
4.2.2.2 Diagrama de secuencia de MESDIF 61
4.2.2.3. Diagrama de secuencia de NMEEF-SD 62
4.3. Implementación y pruebas 63
Lenguaje de programación....63
Entorno de desarrollo....66
Pruebas....67
5. Análisis, diseño e implementación de la interfaz web 72
5.1. Análisis de requerimientos 72
Requerimientos funcionales 72
Requerimientos no funcionales 73
Casos de uso....74
5.2. Diseño....77
Personas....77
5.2.1.1. Persona 1: Carlos Alonso Martínez 77

5.2.1.2. Persona 2: Miguel Álvarez del Castillo 78
Escenarios 79
5.2.2.1. Escenario 1 79
5.2.2.2. Escenario 2 80
5.2.2.3. Escenario 3 80
5.2.2. Storyboard 80
5.3. Implementación y pruebas 83
6. Experimentación y conclusiones finales 86
5.2.3. 6.1. Experimentación con bases de datos públicas 86
Características de la experimentación 86
Resultados obtenidos y análisis 89
6.1.1. 6.1.2.1. Resultados usando representación canónica 89
6.1.2. 6.1.2.2. Resultados para representación DNF 91
6.2. Conclusiones finales y trabajo futuro 92
Conclusiones 92
6.2.1. Trabajo Futuro 93
6.2.2. 7. Bibliografía 96
Anexo I Manual de instalación 101
Instalación de R 101
AI.1.1. Windows 101
AI.1.2. Requisitos previos 101
AI.1.3. Instalación 101
AI.1.4. Ubuntu 105
AI.1.5. Requisitos Previos 105
AI.1.6. Instalación 105
Instalación de RStudio. 107
AI.1.7. Requisitos Previos. 107
AI.1.8. Windows 108
AI.1.9. Ubuntu 109
Instalación del paquete “SDR” 109
Anexo II Manual de Usuario 112
Manual de Uso del paquete. 112
AII.1.1. Funciones disponibles 112
AII.1.2. Uso del paquete 113
Manual de uso de la interfaz 119
AII.1.3. Análisis exploratorio de datos 119

AII.1.4. Ejecución de los algoritmos....122

Índice de figuras

Figura 1.1. Cantidad de datos transferida en 58 s. 12
Figura 1.2. Diagrama de Gantt 19
Figura 1.3. Diagrama de Gantt 19
Figura 2.1. Representanción de conjuntos difusos triangulares....32
Figura 2.2. Intersección (Izqda.) y unión (dcha.) de conjuntos difusos....33
Figura 3.1. Esquema de funcionamiento de SDIGA 38
Figura 3.2. Representación de una regla canónica....39
Figura 3.3. Codificación de una regla DNF....40
Figura 3.4. Esquema de funcionamiento de MESDIF 45
Figura 3.5. Representación del *cuboide* 50
Figura 4.1. Diagrama de casos de uso de la librería....54
Figura 4.2. Diagrama de clases de SDIGA....57
Figura 4.3. Diagrama de clases de MESDIF....58
Figura 4.4. Diagrama de clases de NMEEF-SD....59
Figura 4.5. Diagrama de secuencia de SDIGA....60
Figura 4.6. Diagrama de secuencia de MESDIF....61
Figura 4.7. Diagrama de secuencia de NMEEF-SD....62
Figura 4.8. Funciones usando los diferentes sistemas de POO de R 64
Figura 4.9. Comparativa de tiempo entre sistemas de POO de R 64
Figura 4.10. Funciones para pruebas de eficiencia de R....65
Figura 4.11. Comparativa de tiempos de ejecución de funciones de R....65
Figura 4.12. RStudio 66
Figura 5.1. Diagrama de casos de uso de la interfaz web. 74
Figura 5.2. Carlos Alonso Martínez 77
Figura 5.3. Miguel Álvarez del Castillo....78
Figura 5.4. Storyboard, pantalla #Principal....81
Figura 5.5. Storyboard, pantalla #Resultados....82
Figura 5.6. Aplicación *Shiny* simple....83
Figura 6.1. Esquema de validación cruzada....87
Figura A I.1. Advertencia de seguridad al abrir archivo....102
Figura A I.2. Selección de carpeta de destino de R....103
Figura A I.3. Instalación de componentes adicionales....103
Figura A I.4. Selección de carpeta del Menú de Inicio....104
Figura A I.5. Selección de tareas adiciones en la instalación 104
Figura A I.6. Adición de línea de APT para instalación de R....106
Figura A I.7. Consola de R en línea de comando de Ubuntu....107
Figura A I.8. Selección de carpeta de destinio de RStudio 108
Figura A I.9. Instalación de RStudio desde el Centro de Software de Ubuntu 109
Figura A I.10. Instalación del paquete de R en formato .tar.gz 110
Figura A II.1. Fichero de parámetros para ejecutar MESDIF 114
Figura A II.2. Ejecución de MESDIF en R usando fichero de parámetros. 114
Figura A II.3. Carga en memoria de los conjuntos de entrenamiento y test. 115

Figura A II.4. Cambio de variable objetivo desde la consola de R 115
Figura A II.5. Ejecución de MESDIF a partir del uso de los parámetros de la función.....116
Figura A II.6. Información sobre la ejecución del algoritmo.....117
Figura A II.7. Reglas generadas por el algoritmo.....118
Figura A II.8. Medidas de calidad de las reglas aplicadas al conjunto de test.....118
Figura A II.9. Interfaz con un dataset cargado. 120
Figura A II.10. Interfaz en su estado inicial. 120
Figura A II.11. Visualización de los datos en forma de histograma. 121
Figura A II.12. Parámetros modificables desde la interfaz para el algoritmo SDIGA.....122
Figura A II.13. Visualización de las reglas obtenidas. 123

Indice de tablas

Tabla 1.1 Identificación de tareas.....18
Tabla 1.2. Estimación por líneas de código. 21
Tabla 4.1. Narrativa del caso de uso "Ejecutar SDIGA".....54
Tabla 4.2. Narrativa del caso de uso "Ejecutar MESDIF" 55
Tabla 4.3. Narrativa del caso de uso "Ejecutar NMEEF-SD" 55
Tabla 4.4. Resultados de prueba con el dataset *Iris* para SDIGA 67
Tabla 4.5. Resultados de prueba con el dataset *Iris* para MESDIF.....68
Tabla 4.6. Resultados de prueba con el dataset *Iris* para NMEEF-SD.....68
Tabla 4.7. Resultados de prueba con el dataset *monk-2* para SDIGA 69
Tabla 4.8. Resultados de prueba con el dataset *monk-2* para MESDIF.....69
Tabla 4.9. Resultados de prueba con el dataset monk-2 para NMEEF-SD.....69
Tabla 5.1. Narrativa del caso de uso "Seleccionar Fichero".....75
Tabla 5.2. Narrativa del caso de uso "Seleccionar Algoritmo" 75
Tabla 5.3. Narrativa del caso de uso "Visualizar gráficas y estadísticas".....76
Tabla 6.1. Características de las bases de datos para la experimentación.....86
Tabla 6.2. Parámetros utilizados para SDIGA 87
Tabla 6.3. Parámetros utilizados para MESDIF 88
Tabla 6.4. Parámetros utilizados para NMEEF-SD.....88
Tabla 6.5. Resultados de las experimentaciones utilizando representación canónica 89
Tabla 6.6. Resultados de las experimentaciones utilizando representación DNF 91

Índice de ecuaciones

Ecuación 1 27
Ecuación 2 27
Ecuación 3 28
Ecuación 4 28
Ecuación 5 28
Ecuación 6 28
Ecuación 7 34
Ecuación 8 34
Ecuación 9 42
Ecuación 10. Para soporte Nítido 42

Ecuación 11. Para soporte Difuso 42
Ecuación 12 43
Ecuación 13 46
Ecuación 14 46
Ecuación 15 50

Capítulo 1:

Introducción, definición y planificación del Proyecto

1. Introducción, definición y planificación del Proyecto

1.1. Introducción

A día de hoy, realizamos infinidad de tareas en un ordenador, la mayoría de ellas son a través de Internet y cada vez consumimos más servicios a través de este medio. La posibilidad de tener Internet en cualquier lugar gracias a *tablets* y *smartphones* ha permitido la potenciación del uso de la web para todo tipo de fines. Todas nuestras operaciones que realizamos en Internet (redes sociales, blogs, búsqueda de información…), e incluso las que se realizan *offline* como todas nuestras compras diarias, operaciones bancarias, expedientes médicos, académicos y un largo etcétera de datos son almacenados todos los días en los servidores de los diferentes servicios que utilizamos. Como un pequeño ejemplo, existe una página en la que podemos ver aproximadamente la cantidad de datos transferidos por los gigantes de Internet [1].

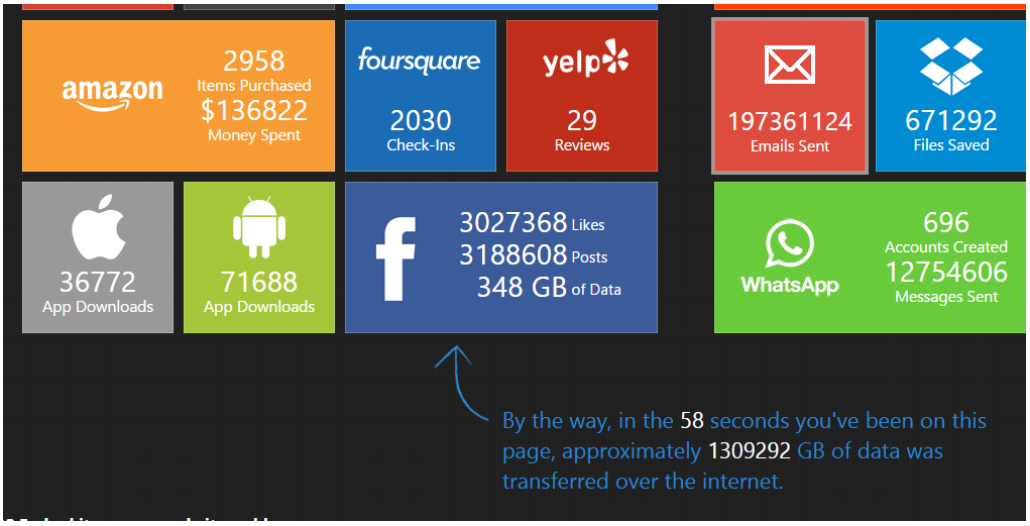


Figura 1.1. Cantidad de datos transferida en 58 s.

Tal y como se aprecia en la Figura 1.1 podemos ver la cantidad de datos que se han transferido en estas páginas de manera aproximada durante 58 segundos. La cantidad de información es enorme (casi 1,5 Millones de GB).

Toda esta cantidad de datos debe de ser analizada profundamente pues contiene gran conocimiento sobre nuestro comportamiento. Esto es interesante para las empresas pues les permite ofrecernos una mejor experiencia con sus productos [2], bien para

estudiar largas secuencias de ADN para el estudio de enfermedades [3] e incluso para analizar nuestras opiniones y sentimientos expresados en las redes [4], entre otras.

Hasta hace muy poco tiempo, todo este análisis de datos se ha realizado a mano. Gracias al abaratamiento de los costes de almacenaje y la imposibilidad de realizar a mano todo este análisis, se empezaron a desarrollar herramientas de análitica de datos como las herramientas OLTP (On-Line Transaction Processing) y OLAP(On-Line Analitical Processsing). Sin embargo, estas herramientas no son lo suficientemente potentes como para extraer **conocimiento** de manera automática o semi-automática como sí lo permite la minería de datos.

La minería de datos trata de extraer conocimiento que no sea trivial, normalmente desconocido y potencialmente útil para el usuario a partir de grandes cantidades de datos [5]. Tradicionalmente esta extracción de conocimiento se ha desarrollado desde dos enfoques claramente diferenciados: un enfoque predictivo, en el cual se pretende predecir el valor de una clase prefijada de antemano por el usuario, y otro enfoque descriptivo en el que se buscan realizar resúmenes de datos o bien buscar relaciones entre los ejemplos de un conjunto de datos.

A pesar de la clara distinción que se realiza entre las diferenes técnicas en minería de datos, existen otras que se encuentran a medio camino entre la minería predictiva y la descriptiva y una de ellas es el llamado “descubrimiento de subgrupos” el cual abordaremos en este proyecto. Esta técnica de minería de datos busca la obtención de subgrupos que sean interesantes en el sentido de que posean una distribución estadística inusual en relación a una propiedad de estos datos en los que el usuario esté interesado. [6]

1.2. Motivación

Como hemos explicado anteriormente, el interés de las empresas por la obtención de este conocimiento hace que a día de hoy se esté generando gran cantidad de puestos de trabajo [7]. Además, el descubrimiento de subgrupos es un campo de la minería de datos relativamente nuevo comparado con el resto de disciplinas. Las características que posee y el éxito que está teniendo en muchas de sus aplicaciones está llamando la atención de muchos investigadores.

Actualmente, existen múltiples plataformas para la utilización de algoritmos de minería de datos, como por ejemplo WEKA [8], KEEL [9] o VIKAMINE [10], entre otras. En este trabajo nuestro objetivo será R [11]. A día de hoy, R es el lenguaje de programación / herramienta estadística más utilizado en ciencia de datos [12] y el número de usuarios va en aumento con cada año que pasa. Sin embargo, en sus repositorios no posee ninguna librería que contenga algoritmos de descubrimiento de subgrupos realizados de manera nativa.

Por lo tanto, este proyecto pretende crear una librería que contenga algoritmos de descubrimiento de subgrupos que se pueda integrar fácilmente en R. El objetivo es facilitar a los investigadores su labor al no tener que cambiar de plataforma para realizar sus experimentaciones y así evitar posibles errores. Pensamos que este proyecto supone un aporte importante a la comunidad científica.

Los algoritmos de descubrimiento de subgrupos que implementaremos serán **SDIGA** [13]**, MESDIF** [14] **y NMEEF-SD** [15]. Estos algoritmos se encuentran desarrollados únicamente para la plataforma KEEL y han tenido éxito en diferentes aplicaciones, principalmente médicas. La característica fundamental de éstos algoritmos es que siguen un enfoque evolutivo. Este enfoque lo consiguen implementando un algoritmo genético como base, lo que les permite obtener soluciones normalmente no óptimas en espacios de búsqueda muy grandes, pero si muy buenas en relación al tiempo de ejecución empleado para descubrir los subgrupos. Destacamos que el tiempo de ejecución de un algoritmo que siga una estrategia de búsqueda exhaustiva se eleva de **manera exponencial** al número de atributos que tengamos. La gran ventaja que poseen frente al resto de algoritmos de

la bibliografía especializada es que permiten trabajar con valores perdidos y hacen uso de lógica difusa para aumentar las interpretabilidad de los subgrupos obtenidos.

Adicionalmente a la creación de la librería con algoritmos de descubrimiento de subgrupos, crearemos una plataforma web sencilla que permita aprovechar todo el potencial que ofrece la librería sin la necesidad de tener R instalado si no se desea. Esta herramienta permitirá la visualización de diferentes propiedades del conjunto de datos para la realización de un análisis exploratorio. El análisis exploratorio es una fase previa a la utilización de los algoritmos de minería de datos muy importante. Así pues, el aporte a la comunidad científica es aún mayor.

1.3. Definición y Planificación del Proyecto

1.3.1. *Definición de los objetivos*

Para la realización del proyecto se definen los siguientes objetivos:

1. Realizar un estudio del estado del arte de la minería de datos en general y de la metodología de descubrimiento de subgrupos en particular.
2. Análisis, diseño e implementación de diferentes algoritmos de descubrimiento de subgrupos en R.
3. Anásis, diseño e implementación de una plataforma web para la utilización de dichos algoritmos.
4. Unificación de todos los algoritmos y la plataforma web en un único paquete de R, para facilitar al máximo su uso e instalación.
5. Realización de experimentaciones con diferentes bases de datos públicas, aplicando los algoritmos implementados.
6. Realización de manuales, ayudas, etc. para la fácil instalación y uso de la librería y sus algoritmos así como de la plataforma web.

1.3.2. *Estructura de la memoria del Proyecto*

A continuación se detalla la estructura de la presente memoria del proyecto:

Este primer capitulo nos introduce en materia y describe las motivaciones por las que realizamos este proyecto, además de detallar los objetivos del mismo, la planificación temporal y el presupuesto.

En el Capítulo 2 se realiza un estudio del estado del arte en minería de datos en el que se explican brevemente sus características, estudiando en mayor profundidad la tarea de descubrimiento de subgrupos. En este capítulo también se justifica la elección de los algoritmos que forman parte de la librería

En el Capítulo 3 se explicará con detalle el funcionamiento de los diferentes algoritmos de descubrimiento de subgrupos implementados.

En el Capítulo 4 se detallan las tareas de ingeniería del software orientadas al análisis, diseño e implementación de nuestra librería de algoritmos, haciendo una breve introducción al lenguaje de programación R.

En el Capítulo 5 se detallan las diferentes tareas de ingeniería del software orientadas al análisis, diseño e implementación de nuestra plataforma web, así como una breve introducción del *framework* Shiny.

En el Capítulo 6 se detallan las experimentaciones realizadas para la validación de los algoritmos. Este capítulo concluye indicando los trabajos futuros relativos a este proyecto.

Por último, se incluyen los anexos relacionados con la instalación y el manual de usuario.

1.4. Agenda y planificación del proyecto

1.4.1. *Planificación del Proyecto*

El proyecto se ha desarrollado entre el 1 de Noviembre de 2014 y el 22 de Junio de 2015. En ese lapso de tiempo hemos empleado en nuestro proyecto 300 horas, que son las establecidas por 12 créditos ECTS.

Es obvio que nuestro proyecto es altamente paralelizable, ya que la implementación de cada algoritmo puede ser realizada por una persona diferente. Sin embargo, la planificación se realizará teniendo en cuenta que el proyecto será llevado a cabo por una sola persona.

Para una correcta planificación es esencial identificar correctamente las tareas que tenemos que realizar, las dependencias que estas tienen entre sí y el tiempo que durarán de la manera más precisa posible ya que así podremos ajustar mejor la agenda y el presupuesto.

Las tareas a realizar en nuestro proyecto serán las siguientes:

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
Estudio del estado del arte en Subgroup Discovery	45 días	sáb 01/11/14	lun 15/12/14	
Estudio de la sintaxis de R y familiarización con la herramienta	45 días	sáb 01/11/14	lun 15/12/14	
Análisis de requerimientos	3 días	lun 15/12/14	jue 18/12/14	2,1
Creación de diagramas de casos de uso	2 sem.	jue 18/12/14	sáb 03/01/15	3
Creación de escenarios	2 días	sáb 03/01/15	lun 05/01/15	4
Modelado de datos	1 sem	lun 05/01/15	mié 14/01/15	5
Creación del storyboard	2 días	mié 14/01/15	vie 16/01/15	6
Diagrama de clases	1 sem	vie 16/01/15	vie 23/01/15	7
Implementación Algoritmo 1	1 ms	vie 23/01/15	dom 22/02/15	8
Pruebas Algoritmo 1	2 días	dom 22/02/15	mar 24/02/15	9
Implementación Algoritmo 2	1 ms	mar 24/02/15	jue 26/03/15	10
Pruebas Algoritmo 2	2 días	jue 26/03/15	sáb 28/03/15	11
Implementación Algoritmo 3	1 ms	sáb 28/03/15	mar 28/04/15	12
Pruebas Algoritmo 3	2 días	mar 28/04/15	jue 30/04/15	13
Creación Plataforma Web	2 sem.	jue 30/04/15	vie 15/05/15	14
Pruebas Plataforma Web	2 días	vie 15/05/15	dom 17/05/15	15
Creación de anexos y manuales	2 sem.	dom 17/05/15	dom 31/05/15	16
Revisión de memoria	2 sem.	dom 31/05/15	dom 14/06/15	17
FIN DE PROYECTO	0 días	dom 14/06/15	dom 14/06/15	18

Tabla 1.1 Identificación de tareas

El diagrama de Gantt resultante de esta tabla de actividades es el siguiente:

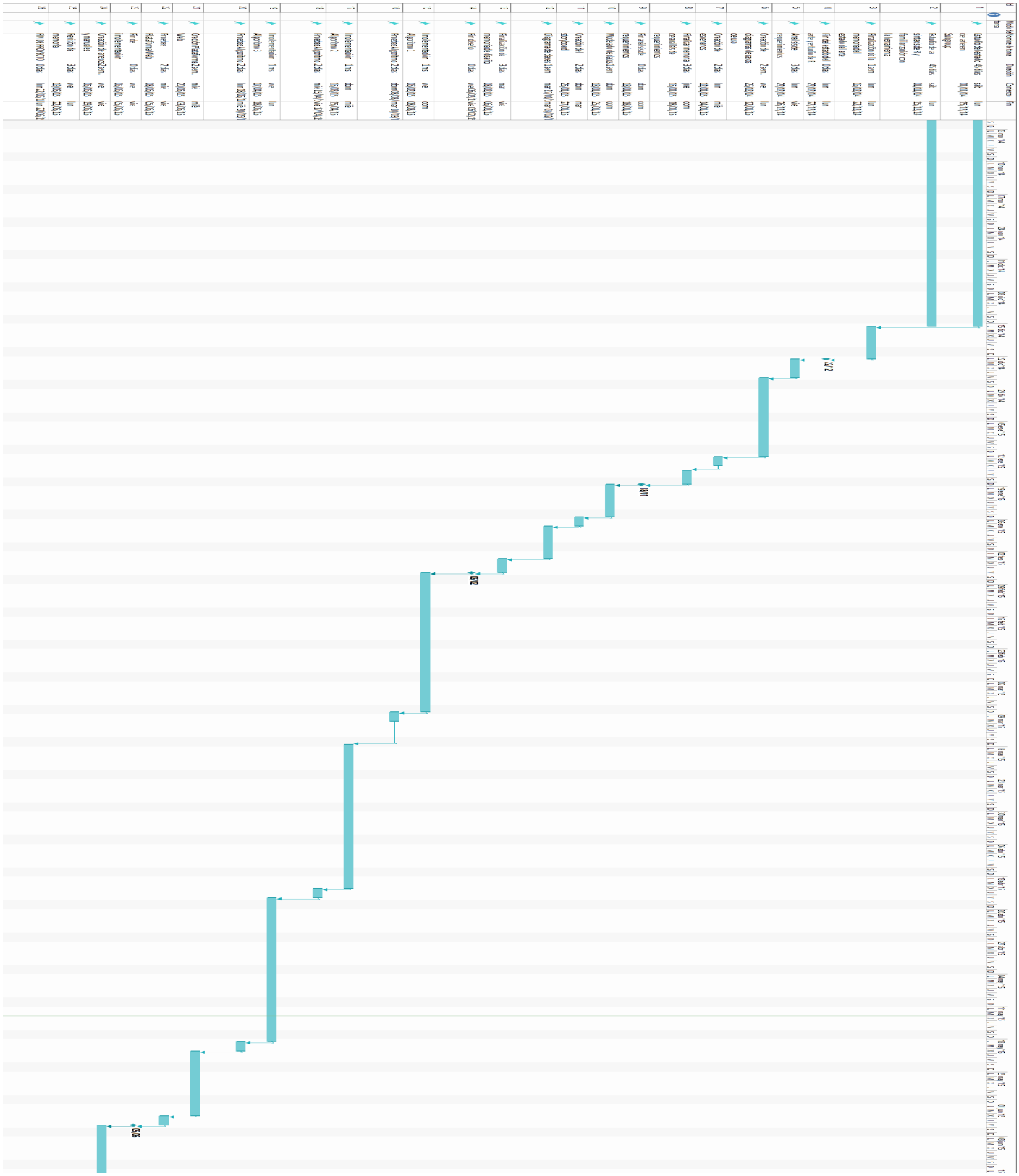


Figura 1.2. Diagrama de Gantt

Estimación de costes

La estimación inicial del coste de un proyecto es una de las partes más importantes del proceso de ingeniería del software, ya que se pretende asignar un presupuesto acorde al esfuerzo a realizar. Así pues debemos evitar la subestimación, pues perderíamos dinero y la sobrestimación pues haríamos que el cliente quedará muy descontento e incluso podría llegar a cancelar el proyecto.

La estimación de un proyecto de software no es sencilla, menos aún en fases iniciales donde los detalles del mismo son muy limitados y es difícil estimar con precisión. Aquí la experiencia juega un papel importantísimo.

Una aproximación habitual para realizar una estimación inicial del presupuesto del proyecto consiste en aplicar el modelo COCOMO II con un estimación por puntos de función. Sin embargo, para la aplicación de este modelo es necesario establecer un valor asociado al lenguaje de programación que se utilice en la implementación, en nuestro caso el lenguaje de programación R. Por desgracia, no hemos podido localizar información respecto al factor que debe utilizarse para el lenguaje R.

De esta forma, realizaremos la estimación utilizando otra técnica, en este caso la estimación por tabla de lineas de codigo, técnica en la que no se necesita utilizar el factor asociado al lenguaje de programación.

Este modo de estimar el coste de un proyecto software consiste en hacer por cada función, una estimación pesimista, una optimista y la más probable de líneas de código, obteniéndose el valor esperado de líneas de código. A continuación, se estima el coste por línea y la productividad del equipo en líneas/mes. La suma del coste de todas las funciones hace el coste total del proyecto. La siguiente tabla recoge los datos correspondientes a la estimación de nuestro proyecto:

Función	Tamaño en líneas de código				€/Línea	Líneas/mes	Coste (€)	Meses
	Optimista	Más probable	Pesimista	Esperado				
Leer dataset	50	120	160	115	0,5	1000	57,50	0,12
Crear Cjtos. Difusos	30	50	75	50,83	0,5	1000	25,42	0,05
Leer parámetros	40	60	100	63	0,5	1000	31,5	0,06
Mostrar resultados	40	60	80	60	0,5	3000	30	0,02
Genético SDIGA	300	450	600	450	3	400	1350	1,13
Busqueda Local SDIGA	30	60	90	60	1,5	400	90	0,15
Genético MESDIF	300	500	800	516	4	400	2064	1,29
Genético NMEEF-SD	300	500	800	516	4	400	2064	1,29
Testeo de reglas	30	45	70	46	2	600	92	0,08
Interfaz Web	100	200	400	216	2	800	432	0,27
Servidor Web	450	700	1000	708	35	1200	2478	0,59
TOTAL	**1670**	**2745**	**4175**	**2800,83**	**22**	**10200**	**8714,42**	**5,05**

Tabla 1.2. Estimación por líneas de código.

A este presupuesto deberemos de sumarle el coste de la amortización de nuestros equipos así como el uso de software de pago que tengamos.

Nuestro equipo de desarrollo es un ordenador de sobremesa con las siguientes características:

- Intel® Xeon® CPU E31230 @ 3.20GHz × 8
- 16 GB de RAM
- Tarjeta Gráfica NVidia GeForce 8600 GTS
- 500 GB de disco duro
- Sistema operativo Elemantary OS Luna (basada en Ubuntu 12.04)

Todas nuestras herramientas, a excepción del hardware y la suite Office, son bajo licencia GPL, por lo que no nos ha significado coste alguno. No obstante, tenemos la restricción de que nuestro software también sea bajo licencia GPL.

En cualquier caso, suponemos una amortización del equipo durante los 5 meses de trabajo de 145.83 € (Coste del equipo 3500 € aprox. a amortizar en 10 años).

Por lo tanto, el costo total del proyecto asciende a :

8714.41 + 145.83 = **8860.24 €**

Capítulo 2:
Introduccion a la minería de datos

2. Introducción a la minería de datos

2.1. Minería de datos

La minería de datos trata de extraer patrones que no sean triviales, normalmente desconocidos y potencialmente útiles para el usuario, a partir de grandes cantidades de datos [5].

La minería de datos es el paso más importante de un proceso denominado *Knowledge Discovery in Databases* (KDD). El *KDD* tiene como finalidad extraer conocimiento de un gran volumen de datos. Estos datos pueden provenir de diversas fuentes o no. Un ejemplo puede ser la obtención de reglas que indiquen qué tipo de pacientes pueden visitar una franja horaria en un servicio de urgencias [16].

Como se ha indicado antes, dentro de la minería de datos nos encontramos con dos posibles modelos de aplicación de la misma:

- Minería de datos predictiva: Cuyo objetivo es intentar predecir el valor de una variable en nuevas instancias.
- Minería de datos descriptiva: Cuyo objetivo es descubrir relaciones entre los datos actuales.

Debido a la naturaleza de cada modelo, los métodos utilizados y las aplicaciones son diferentes. Así por ejemplo podemos utilizar la minería de datos predictiva para problemas de regresión y clasificación, con métodos como las redes neuronales y algoritmos de árboles de decisión como C4.5 [17]. Igualmente se puede utilizar la minería de datos descriptiva para problemas de resumen y agrupamiento de datos o bien para la obtención de reglas de asociación, con algoritmos como KMeans [18] o Apriori [19] respectivamente.

2.2. Descubrimiento de subgrupos

A pesar de la distinción realizada anteriormente, existen tareas en minería de datos que se encuentran a medio camino entre ambos enfoques y que por lo tanto, poseen características tanto descriptivas como predictivas. Este es el caso del denominado "*descubrimiento de subgrupos"*. Una definición de la tarea puede ser la presentada en [6]:

"*Partiendo de una población de individuos y de una propiedad de estos en la que estamos interesados, la tarea de descripción de subgrupos puede ser definida como la obtención de subgrupos de individuos de la población con propiedades estadísticamente "interesantes", por ejemplo, son lo más grandes posibles y tienen las características de distribución estadística más inusuales para la propiedad de interés*."

Según esta definición, el descubrimiento de subgrupos trata de buscar relaciones inusuales e interesantes entre los datos teniendo prefijado un valor de una variable objetivo. Por ejemplo, si queremos saber qué relaciones tienen los pacientes que han sufrido diabetes tipo dos, tendríamos que utilizar este modelo, ya que es capaz de encontrar relaciones interesantes (estadísticamente hablando) entre estos pacientes.

Para poder representar el conocimiento generado, normalmente se utilizan reglas del tipo:

Si (Condición) → *VariableObjetivo*

Donde la parte del consecuente solo posee una única variable, la variable objetivo.

No podemos encuadrar la tarea en ningún modelo anterior puesto que realiza un aprendizaje supervisado (tenemos los ejemplos etiquetados por una variable objetivo que los clasifica) para obtener una descripción de los datos, donde normalmente se realiza un aprendizaje no supervisado en el cual no existe una etiqueta de clase para cada ejemplo.

Un punto a destacar de este modelo es su diferencia con la predicción. Las reglas que se generan en descubrimiento de subgrupos no son utilizadas para predecir, sino para describir el comportamiento de los ejemplos del conjunto de datos en relación a un valor de la variable objetivo. Por lo tanto, para poder cubrir todos los valores de la variable objetivo, es necesaria una ejecución del algoritmo por cada valor. Además dichos modelos, al tener objetivos diferentes (predecir en minería de datos predictiva y describir en descubrimiento de subgrupos) tanto los métodos utilizados como las medidas de calidad utilizadas son diferentes.

Los métodos que se encuadren dentro de este modelo de minería de datos deben de tener las siguientes características:

- Tipo de variable objetivo: Las variables objetivo pueden ser de tipo binario (poseen solo dos valores), nominales (poseen *n* valores), o bien numéricas (la variable puede contener valores dentro de un rango numérico). En nuestro proyecto, nos centraremos solo en variables objetivo de tipo nominal y binario.
- Lenguaje de descripción: Las reglas deben ser lo más sencillas posible y por ello normalmente se utilizan reglas compuestas por conjunciones de pares atributo-valor o bien en forma normal disyuntiva. También, para intentar aumentar la interpretabilidad, se puede utilizar lógica difusa.
- Medidas de calidad: Son muy importantes ya que definirán cuán “interesante” es un subgrupo. Hablaremos de ellas a continuación.
- Estrategia de búsqueda: Como se ha comentado anteriormente, el espacio de búsqueda aumenta de manera exponencial al número de atributos. En este sentido, la utilización de metaheurísticas que buscan reducir el espacio de búsqueda para intentar encontrar buenas soluciones es vital.

2.3. Medidas de calidad para el descubrimiento de subgrupos

Las medidas de calidad en el descubrimiento de subgrupos tienen una doble función: permiten por un lado mostrar al experto información sobre los subgrupos obtenidos y por otro permiten generar dichos subgrupos, ya que definen cómo de "interesante" es un subgrupo.

A día de hoy no existe ningún consenso entre los expertos en qué medida de calidad es mejor para la tarea, pero sí se definen muchas medidas que pueden ser utilizadas y comprobar resultados con ellas. Dichas medidas son:

- Número de reglas (N_r): Indica el número de subgrupos generados.
- Número de variables (N_v): Determina el número medio de variables que hay en las reglas generadas.
- Soporte: Debido a que las reglas generadas son descriptivas, esta medida es la misma que en minería de datos descriptiva:

$$Sup(x) = \frac{n(Cond \wedge T_v)}{n_s}$$

Ecuación 1

Donde $n(Cond \wedge T_v)$ indica el número de ejemplos correctamente cubiertos, es decir, que cumplen el antecedente y tienen como consecuente nuestra clase objetivo (T_v). n_s es el número total de ejemplos en nuestro conjunto de datos.

- Cobertura (*Coverage*): Muestra el porcentaje de ejemplos que son cubiertos por la regla en relación al total de ejemplos:

$$Cov(x) = \frac{n(Cond)}{ns}$$

Ecuación 2

Donde $n(Cond)$ alude a los ejemplos que cumplen únicamente el antecedente de la regla.

- Confianza: Esta medida revela el porcentaje de ejemplos correctamente cubiertos del total de ejemplos cubiertos:

$$Conf(x) = \frac{n(Cond \wedge T_v)}{n(Cond)}$$

Ecuación 3

- Interés: Mide el interés de una regla:

$$Int(x) = \frac{\sum_{i=1}^{n_v} Gain(A_i)}{n_v \cdot log_2(|dom(G_k)|}$$

Ecuación 4

Donde *Gain* es la ganancia de información, A_i es el número de valores de la variable y $|dom(G_k)|$ es la cardinalidad de la variable objetivo.

- Significancia: Esta medida refleja la novedad en la distribución de los ejemplos cubiertos por la regla respecto al conjunto completo de ejemplos:

$$Sig(x) = 2 \cdot \sum_{k=1}^{n_c} n(Cond \wedge T_{vk}) \cdot \log\left(\frac{n(Cond \wedge T_{vk})}{n(Cond \wedge T_v) \cdot p(Cond)}\right)$$

Ecuación 5

- Atipicidad (*Unusualness*): Se define como la ponderación de la precisión relativa de la regla intentando establecer un equilibrio entre cobertura y precisión:

$$WRAcc(x) = \frac{n(Cond)}{n_s}\left(\frac{n(Cond \wedge T_v)}{n(Cond)} - \frac{n(T_v)}{n_s}\right)$$

Ecuación 6

Donde $n(T_v)$ es el número de ejemplos de la varible objetivo.

2.4. Principales algoritmos de descubrimiento de subgrupos

Para abordar la tarea de descubirmento de subgrupos se ha desarrollado un conjunto amplio de algoritmos. La mayoría de ellos son extensiones de algoritmos clásicos de minería de datos de clasificación o de reglas de asociación, adaptados para esta tarea. A continuación se detalla una clasificación de los distintos algoritmos de descubrimiento de subgrupos existentes [20]:

2.4.1. *Extensiones de algoritmos de clasificación.*

Los primeros algoritmos de descubrimiento de subgrupos fueron EXPLORA y MIDOS que se implementaron como extensiones de clasificadores para realizar la tarea de descubrimiento de subgrupos.

- EXPLORA [21] fue el primer enfoque desarrollado para descubrimiento de subgrupos. Utiliza árboles de decision para extraer reglas descriptivas. Para medir el interés de una regla utiliza medidas estadísticas. El método de búsqueda aplicado puede ser heurístico o exhaustivo, pero sin realizar poda.
- MIDOS [22] amplía EXPLORA para su utilización en bases de datos multirelacionales. También busca de manera exhaustiva o heurística, pero utilizando poda basada en soporte mínimo para reducir el espacio de búsqueda.

2.4.2. *Algoritmos basados en clasificación*

- SD [23]: Es un sistema de basado en una variación de la búsqueda por haz y está guiada por conocimiento experto. En vez de definir una medida óptima para elegir automáticamente el mejor subgrupo, define una serie de ayudas para guiar al experto en la realización de búsquedas flexibles y efectivas. Los subgrupos generados deben superar un umbral mínimo de soporte, intentándose buscar subgrupos que cubran el mayor número posible de ejemplos del valor objetivo y el mínimo número de ejemplos que no son del valor objetivo.

- CN2-SD [24]: Este algoritmo es una adaptación del clasificador CN2 [25] para la tarea de descubrimiento de subgrupos. Para seleccionar subgrupos la medida que utiliza es una modificación de *unusualness*.

Extensión de algoritmos de reglas de asociación

- Apriori-SD [26]: Es una extensión del clasificador Apriori-C [27] que, a su vez, es una modificación de APRIORI para clasificación. Utiliza un mecanismo de post-procesamiento y *unusualness* como medida de calidad para obtener subgrupos.
- SD-Map [28]: Es un algoritmo exhaustivo de descubrimiento de subgrupos que se basa en el método FP-growth [29]. Utiliza un paso modificado en FP-growth que permite calcular la calidad de un subgrupo directamente sin referenciar a resultados previos. Puede usar muchas medidas de calidad, entre las que destaca *unusualness*.

2.4.3.

2.4.4. *Algoritmos evolutivos*

- SDIGA [13]: Utiliza un algoritmo genético como nucleo, hace uso de lógica difusa para tratar mejor con valores numéricos y permite utilizar diversas medidas para calcular la calidad de los individuos haciendo una ponderación entre ellas. Aplica un esquema iterativo para la extracción de reglas descriptivas difusas.
- MESDIF [30]: Emplea un algoritmo genético multiobjetivo para la extracción de reglas descriptivas difusas. Está basado en el enfoque SPEA2 [31] que aplica conceptos de elitismo en la selección y búsqueda de soluciones en el frente de Pareto. Permite también el uso de gran diversidad de medidas de calidad.
- NMEEF-SD [15]: Usa un algoritmo genético multiobjetivo. Éste utiliza el enfoque NSGA-II [32] cuya estrategia de búsqueda se basa en la ordenación por dominancia y el uso de elitismo. Hace uso de operadores específicos para promover la diversidad.

Como se ha explicado anteriormente implementaremos estos tres últimos algoritmos basados en modelos evolutivos. A pesar de que estos algoritmos no obtienen, normalmente, una solución óptima global, son capaces de obtener una lo suficientemente buena para tener un balance alto en cuanto a calidad de la solución y tiempo de ejecución. Además, estos algoritmos evolutivos se encuadran dentro de los llamados "*Evolutionary Fuzzy Systems*" (EFS) los cuales hacen uso de lógica difusa para trabajar de manera más simple con datos contínuos y aumentar la interpretabilidad de las reglas generadas, algo vital en minería de datos.

2.5. Lógica Difusa

En minería de datos, la comprensión de los modelos generados es un factor crucial en modelos interpretables. La utilización de lógica difusa permite una interpretación natural de variables que son de tipo numérico asignando una serie de etiquetas lingüisticas a un rango de valores.

La lógica difusa nos permite trabajar con dicha información subjetiva realizando una extensión de los conjuntos nítidos, en donde un elemento está (1) o no está (0) en un conjunto. Sin embargo, en lógica difusa, un elemento puede tener valores de pertenencia a un conjunto en el rango [0,1] con 0 indicando no pertenencia y 1 pertenencia total. Por ejemplo, teniendo la representación de los conjuntos difusos de la Figura 2.1 una persona puede tener un grado de pertenencia al conjunto "Alto" de 0.41 y de 0.59 al conjunto "Medio" refiriéndonos a la cualidad "Estatura" (representado como un punto en el plano).

Un punto a destacar de este ejemplo es que el grado de pertenencia a un conjunto difuso no tiene el mismo significado que el concepto de probabilidad. Este último se refiere a *"La probabilidad de que la persona sea Alta es de 0,41",* es decir, se supone que la persona es alta (1) o no (0) teniendo un 41 % de posibilidades de acertar en la afirmación. En el caso de la lógica difusa, nos referimos a que la persona es "más o menos" alta en otros términos, con un valor que corresponde al 0.41.

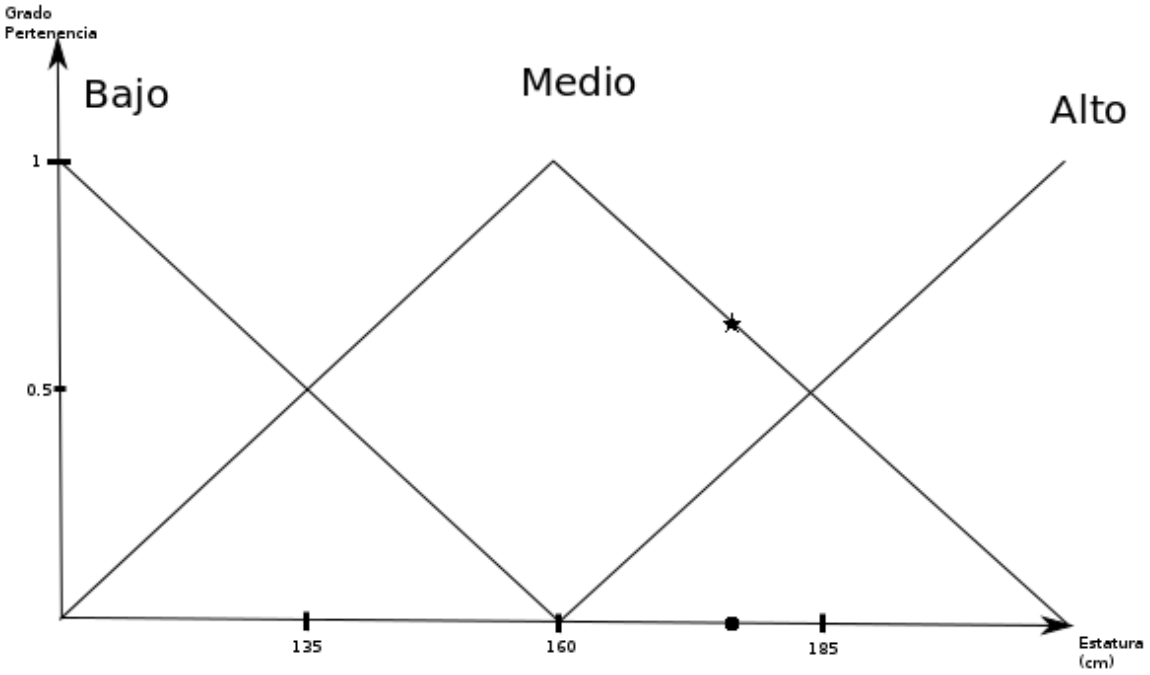


Figura 2.1. Representanción de conjuntos difusos triangulares

Al ser los conjuntos difusos una extensión de los conjuntos nítidos, las operaciones que se pueden realizar sobre estos son las mismas, es decir, se pueden unir, intersecar y negar conjuntos difusos. En nuestro caso de obtención de reglas, si las variables están unidas por conjunciones, el grado de pertenencia se puede calcular con la intersección. Esta se realiza con una t-norma. Esta t-norma puede tener diferentes expresiones, pero las más utilizadas son la t-norma mínima y la t-norma producto. Para una representación DNF (Sección 3.1.2.2) una variable puede contener diferentes valores unidos por una disyunción. El cálculo del grado de pertenencia de un ejemplo a una variable de la regla DNF se realiza usando la unión difusa a través de la t-conorma, que normalmente es el valor máximo.

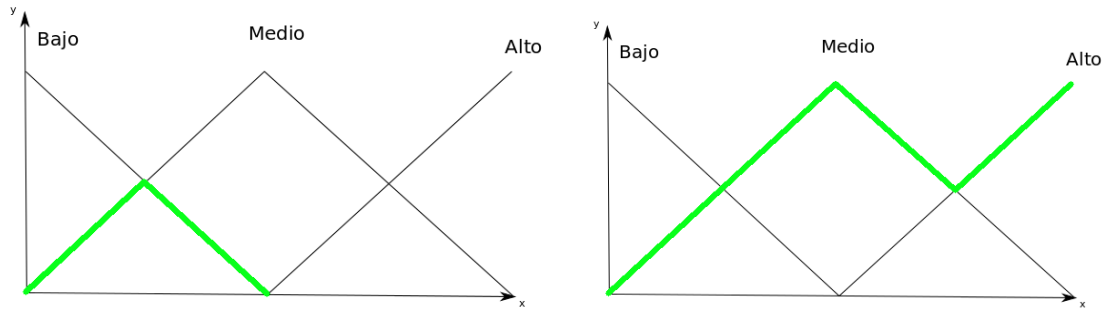


Figura 2.2. Intersección (Izqda.) y unión (dcha.) de conjuntos difusos

La utilización de lógica difusa permite simplificar la dificultad de trabajar con variables de tipo numérico, aumentando la comprensibilidad de las reglas obtenidas. Por lo tanto, el enfoque de los diferentes algoritmos que implementaremos hará uso de la lógica difusa cuando sea necesario.

Las reglas descriptivas de descubrimiento de subgrupos utilizan como base un conjunto de ejemplos $E = \{ e_1, e_2, \dots , e_n\}$ de entrenamiento. Estos ejemplos, a su vez, se componen de un conjunto de variables, que pueden ser categóricas o numéricas y un valor para la clase objetivo.

$$e_n = \{ V_1, V_2, \dots , V_n, clase_i\}$$

Cada una de las variables del ejemplo se compone de un número finito de valores en caso de variables categóricas, o de etiquetas difusas en caso de variables contínuas.

$$V_n = \{C_1, C_2, \dots, C_m\} \text{ o } V_n = \{LL_1, LL_2, \dots , LL_m\}$$

Donde C_m es el valor categórico m para una variable categórica o LL_m es la etiqueta difusa m para una variable numérica.

Para poder calcular las medidas de calidad mencionas en la Sección 2.3, es necesario averiguar, dada una regla, qué ejemplos están cubiertos y en que medida.

Así pues, dado un ejemplo y dada una regla, el grado de pertenencia de un ejemplo a dicha regla viene dado por la expresión *APC (Antecedent Part Compatiblility):*

$$APC(e, R_i) = T(TC\left(\mu_1^1(e_1), \dots \mu_n^1(e_i)\right), \dots \ , TC\left(\mu_1^i(e_1), \dots \mu_n^i(e_i)\right))$$

Ecuación 7

Donde:

- e_k indica el valor del ejemplo para la variable *k*.
- μ^i_n indica el grado de pertenencia al valor *n* de la variable *i.*
- *TC* es la t-conorma difusa. En este caso es la t-conorma máximo.
- *T* es la t-norma difusa. En este caso es la t-norma mínimo.

En caso de que la variable *i* sea categórica, el valor μ^i_n será o uno o cero dependiendo si tiene el mismo valor que el que indique la regla. En caso de ser numérica, μ^i_n se calculará según la función de pertenencia triangular:

$$\mu_{a,b,c}(x) = \begin{cases} 0, & si\ x \leq a \\ \dfrac{x-a}{b-a}, & si\ a \leq x \leq b \\ \dfrac{c-x}{c-b}, & si\ b \leq x \leq c \\ 0, & si\ c \leq x \end{cases}$$

Ecuación 8

Donde *a,b,c* definen el valor inferior, el punto intermedio y el valor superior del conjunto difuso, respectivamente.

Con esta definición, un ejemplo está cubierto por una regla si su valor APC con respecto a dicha regla es mayor a cero.

2.6. Estudio de librerías y plataformas existentes para el descubrimiento de subgrupos

Como se indicó en la introducción, no existe para R ninguna librería que trate la tarea de descubrimiento de subgrupos de manera nativa. Existe un paquete denominado "*rsubgroup*"[1] el cual implementa una interfaz de R para utilizar la herramienta de minería de datos VIKAMINE [10]. La principal diferencia con nuestra librería es que *rsubgroup* es una pasarela para utilizar la herramienta VIKAMINE desde R, mientras que nuestra librería está implementada directamente en R. Por lo que no tenemos dependencias con lenguajes y herramientas externas. Además, los algoritmos implementados en nuestra librería no se encuentran en VIKAMINE.

Los algoritmos que se han realizado se encuentran implementados en Java y están integrados en la herramienta de minería de datos de código abierto KEEL [9]. Por lo que tomaremos como referencia el código fuente de estos algoritmos para realizar el nuestro.

Cabe destacar que la implementación de estos algoritmos en R no es tan sencilla como realizar una simple transcripción de la implementación en Java debido a las características particulares que tiene el lenguaje R. Estas peculiaridades se explican con más detalle en la Sección 4.3.1

[1] http://cran.r-project.org/web/packages/rsubgroup/index.html

Capítulo 3:

Algoritmos de extracción de reglas de descubrimiento de subgrupos

3. Algoritmos de extracción de reglas de descubrimiento de subgrupos

En este capítulo describimos con gran nivel de detalle el funcionamiento de los algoritmos que se han implementado. Como se ha explicado anteriormente, los algoritmos elegidos han sido **SDIGA, MESDIF** y **NMEEF-SD**. A continuación, exponemos sus caracteres

3.1. Algoritmo SDIGA (Subgroup Discovery Iterative Genetic Algorithm)

El algoritmo SDIGA es un algoritmo iterativo de extracción de reglas descriptivas que tiene como núcleo un algoritmo genético. Este algoritmo genético es el encargado de obtener una única regla que tenga un soporte y una confianza mínima preestablecida y que cubra el mayor número de ejemplos posible de la variable objetivo.

Este proceso, al ser iterativo, se ejecuta hasta que se cumple cierta condición de parada. Dicha condición será que la regla generada por el algoritmo genético no sea capaz de cubrir ejemplos no cubiertos por reglas anteriores (ejemplos nuevos), o que dicha regla no cumpla un nivel de confianza mínimo.

A continuación se realiza una optimización local basada en ascensión de colinas, que el usuario puede elegir si utilizar o no, para intentar que dicha regla sea más general aún, es decir, tenga mayor soporte, pero sin perder confianza.

El funcionamiento del algoritmo es el siguiente:

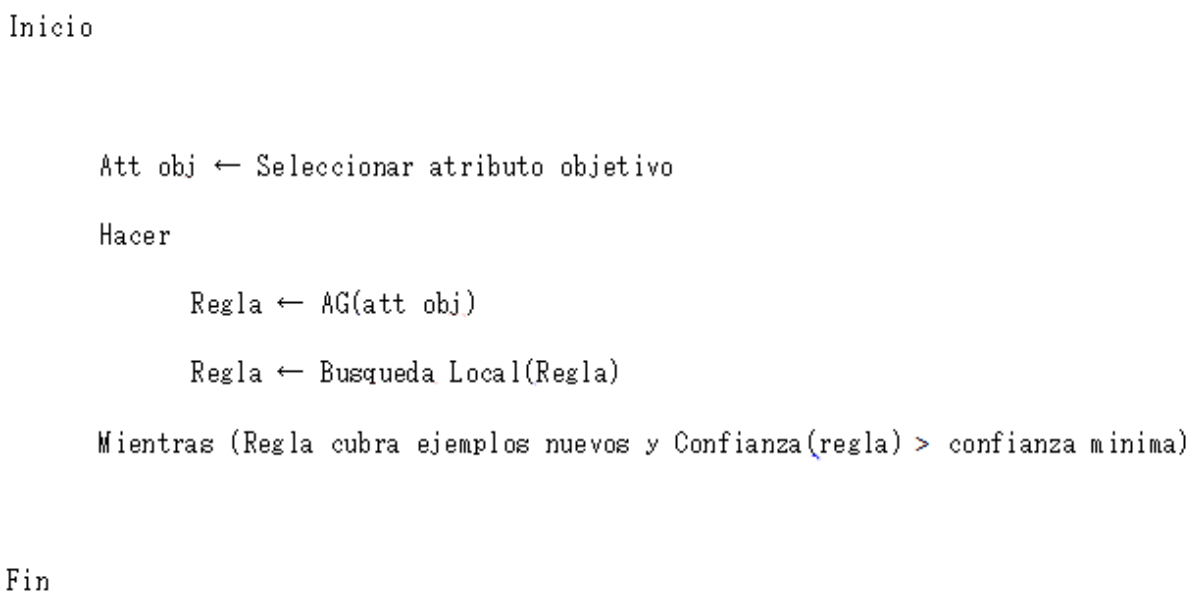


Figura 3.1. Esquema de funcionamiento de SDIGA

SDIGA es capaz de trabajar con valores perdidos, con datos categóricos y contínuos. Para datos contínuos hace uso de lógica difusa para una mayor comprensión de las reglas generadas.

Sin embargo, para la variable de clase, solo se pueden utilizar datos categóricos.

3.1.1. *Funcionamiento del algoritmo genético de SDIGA*

Como hemos explicado, el núcleo de SDIGA es un algoritmo genético, el cual en cada ejecución devuelve una regla. Este algoritmo genético opera con datos númericos utilizando lógica difusa para intentar maximizar la interpretabilidad de las reglas generadas, algo fundamental en minería de datos.

3.1.2. *Esquema de representación*

Cada individuo que conforma la población representa una posible regla. Al tratarse de una tarea de descubrimiento de subgrupos no es necesario incluir el consecuente en la representación, pues ya lo tenemos predefinido.

La solución que nos presenta [33] puede operar con dos representaciones diferentes en función de nuestras necesidades: reglas **canónicas** o reglas **DNF.**

3.1.2.1. Reglas canónicas

Una regla canónica está formada por un antecedente que es una conjunción de pares variable – valor y por un consecuente que es un valor categórico.

Por ejemplo:

$$SI\ X_1 = Valor_1\ \&\ X_3 = LL^1_3\ Entonces\ Clase_2$$

Donde X_1 es una variable categórica que tiene el valor 1 (de tres posibles) de dicha variable y X_3 es una variable numérica que tiene como valor la etiqueta difusa número 1 para la variable 3 (de tres etiquetas difusas creadas). En esta regla teníamos cuatro variables posibles, por lo tanto ni la variable X_2 ni la variable X_4 participan en la regla (ambas son variables categóricas con tres valores posibles). Para representar esta situación, todos los individuos tienen una longitud fija y en el caso de tener variables que no participen en la regla, estas contendrán un valor especial que será el número máximo de variables/etiquetas difusas posibles más uno. En el ejemplo dicho valor es cuatro para todas las variables. La representación del ejemplo se muestra en la siguiente Ilustración:

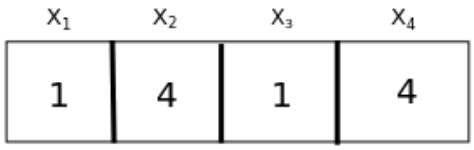


Figura 3.2. Representación de una regla canónica

3.1.2.2. Reglas DNF

Las reglas en formato DNF, o forma normal disyuntiva, son reglas que permiten que una variable pueda utilizar más de un valor en el antecedente de la regla.

Los valores que pertenezcan a la misma variable están unidos por una disyunción y la unión de todas las variables para obtener el antecedente completo se realiza mediante una conjunción. Un ejemplo de regla DNF sería el siguiente:

Si X1 = (Valor1 o Valor3) & X3 = Valor1 Entonces Clase2

Para poder representar esta situación, el esquema anterior no nos sirve. No obstante, si utilizamos un esquema de representación binario, cuya longitud es igual a la suma de todos los posibles valores categóricos o etiquetas difusas de todas las variables es posible representarla sin problemas.

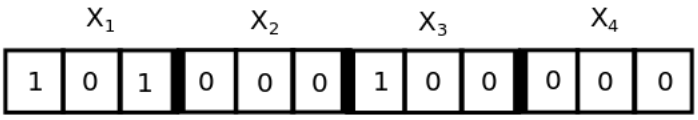


Figura 3.3. Codificación de una regla DNF

Con esta representación, un valor de una variable participa en la regla si este tiene un valor de uno. Asimismo, una variable no participará en la regla si todos sus valores son cero o uno.

3.1.2.3. Operadores genéticos

Los operadores genéticos son los que permiten que el proceso evolutivo avance. Estos operadores, entre otras cosas, permiten que se generen cambios aleatorios en las variables o la generación de nuevos individuos a través de la unión de otros individuos.

3.1.2.4. Operador de cruce

El algoritmo genético de SDIGA utiliza un esquema estacionario puro, en el cual solamente se cruzan los dos mejores individuos de la población. Esto crea una alta presión selectiva y una convergencia rápida a una solución que normalmente es un óptimo local. El operador de cruce que se utiliza es un **operador de cruce en dos puntos.**

3.1.2.5. Operador de mutación

La operación de mutación se lleva a cabo en SDIGA de manera especial. La mutación se realiza sobre cualquier cromosoma de la población de padres o descendientes indistintamente según la probabilidad de mutación. En un esquema clásico, la mutación solo se realiza en la población de descendientes.

El operador de mutación se puede aplicar de dos formas:

- Eliminando una variable aleatoria. Esto se consigue colocando su valor con el valor de no participación para reglas canónicas o todos sus valores a cero para reglas DNF.
- Asignando un valor aleatorio a la variable para reglas canónicas o en todos los valores de dicha variable para reglas DNF. Hay que destacar que el valor de no participación también forma parte de estos posibles valores.

Se aplicará un operador u otro de forma aleatoria, ambos con la misma probabilidad.

3.1.2.6. Función de evaluación

Para calcular la calidad de un individuo de la población es necesario evaluarlo con alguna función. SDIGA es un algoritmo monobjetivo pero busca máximizar más de una medida de calidad (soporte y confianza). Para aunar ambos objetivos en una

única función SDIGA utiliza una media ponderada de ambos objetivos, dejando el cálculo de los pesos para cada objetivo al experto.

Por lo tanto, la función objetivo a maximizar será:

$$f(x) = \frac{Sop(x) * w_1 + Conf(x) * w_2 + Obj3(x) * w_3}{\sum_{i=1}^{3} w_i}$$

Ecuación 9

Donde:

- *Sop(x)* es el soporte local. Este soporte local es una modificación del soporte definido anteriormente en la Sección 2.3 y puede ser nítido o difuso según el usuario desee y se calcula de la siguiente manera:
  - $Sop_n(x) = \frac{Ne^+(R_i)}{Ne_{NC}}$

Ecuación 10. Para soporte Nítido

  - $Sop_d(x) = \frac{\sum_k APC(E^k, R_i)}{Ne_{NC}}$

Ecuación 11. Para soporte Difuso

Donde:

- $Ne^+(Ri)$ indica el número de ejemplos cubiertos por la regla *i* que cumplen el consecuente (ejemplos correctamente cubiertos) y que además no han sido cubiertos por ninguna regla anterior.
- Ne_{NC} denota el número de ejemplos de la clase objetivo que quedan por cubrir.
- *APC* es la expresión *Antecedent Part Compatibility* y sólo se calcula con los ejemplos que no estaban cubiertos por reglas anteriores.

- *Conf(x)* es la confianza tal y como la hemos definido en la Sección 2.3 para el caso nítido. No obstante, para el uso de la confianza difusa la expresión viene

definida como el ratio de la suma de la expresión *APC* de los ejemplos correctamente cubiertos entre la suma de la expresión *APC* de los ejemplos cubiertos por la regla, es decir:

$$Conf_d(x) = \frac{\sum_{E^{cc}} APC(E^k, R_i)}{\sum_{E^c} APC(E^k, R_i)}$$

Ecuación 12

- Donde:
  - E^{cc} son los ejemplos correctamente cubiertos
  - E^c son los ejemplos cubiertos

- *Obj3* es un objetivo adicional que se puede elegir libremente de entre los objetivos definidos en la Sección 2.3
- W_i es el peso del objetivo i

3.1.2.7. Operador de reemplazo

El reemplazo de la población inicial se lleva a cabo uniendo la población de descendientes y la población inicial. El operador ordena todo el conjunto por valor fitness, quedándose con los *n* mejores, siendo *n* el tamaño de la población inicial.

3.1.3. *Optimización Local*

Una vez haya finalizado el algoritmo genético se obtiene la mejor regla de la población. Después, si el usuario así lo desea, se puede realizar un proceso de optimización local, para intentar aumentar el soporte de la regla.

Este proceso de optimización local se lleva a cabo utilizando un algoritmo de ascensión de colinas del primer mejor. Un individuo sera tildado de “mejor” si, al eliminar una de sus variables, su soporte aumenta y su confianza se mantiene igual o aumenta.

3.2. Algoritmo MESDIF (Multiobjetive Evolutionary Subgroup Discovery Fuzzy rules)

MESDIF es un algoritmo genético que extrae reglas difusas cuya representación puede ser canónica o en forma normal disyuntiva.

Este algoritmo sigue el enfoque SPEA2 [31], en el que se utiliza una población élite desde donde se realiza todo el proceso evolutivo. Al final del proceso, las reglas que serán devueltas son las que estén en la población élite.

Como se trata de un algoritmo multiobjetivo, hay que unificar todos los objetivos individuales de alguna manera para intentar maximizarlos (o minimizarlos), sin penalizar de ninguna manera la mejora de un objetivo en otro distinto. Para conseguir esto MESDIF, al igual que SPEA2, utiliza el concepto de elitismo y para rellenar esta población élite, de tamaño fijo, se utiliza un esquema de nichos basado en la dominancia de Pareto.

Un individuo i domina a otro j si y solo si **todos** los valores objetivo de i son iguales o mejores que los valores objetivos de j. En el caso de MESDIF, todos deberán serán mejores o iguales y, al menos uno de ellos, estrictamente mejor.

Este sistema permite encontrar individuos no dominados para después rellenar la población élite con dichos individuos. Además, incorpora una función de truncado si tenemos más individuos no dominados que el tamaño de la población élite y una función de rellenado si tenemos el caso contrario.

3.2.1. *Funcionamiento del algoritmo genético de MESDIF*

MESDIF tiene su base en un algoritmo genético. Este posee una población inicial y una población élite donde se almacenan los individuos no dominados. Una vez se tiene la población élite, el proceso evolutivo continúa solo con la población élite. El esquema general del algoritmo es el siguiente:

Paso 1. Inicialización:
  Generar una población inicial de reglas P_0 y crear una población elite vacía $P'_0 = \emptyset$.
  Establecer la generación inicial (t = 0).
Repetir
  Paso 2. Asignación de valor de adaptación:
    Calcular los valores de adaptación de las reglas de P_t y P'_t.
  Paso 3. Selección de entorno:
    Copiar todas las reglas no dominadas de P_t y P'_t a P'_{t+1}. Utilizar la función de truncado o la de rellenado según sea necesario para obtener el tamaño fijo y preestablecido de P'_{t+1} (N).
  Paso 4. Reproducción:
    Realizar selección por torneo binario con reemplazo sobre P'_{t+1} aplicando después los operadores de cruce en dos puntos y mutación uniforme sesgada, obteniendo P_{t+1}.
  Paso 5. Pasar a la siguiente generación (t = t+1)
Mientras no se alcance el número de evaluaciones de reglas.
Paso 6.
  Devolver todas las reglas no dominadas de P'_{t+1}.

Figura 3.4. Esquema de funcionamiento de MESDIF

3.2.2. *Esquema de representación*

El esquema de representación es el mismo que el utilizado en SDIGA, con reglas canónicas (sección 3.1.2.1) o reglas DNF (sección 3.1.2.2) en función de las necesidades del experto.

3.2.3. *Generación de la población inicial*

MESDIF permite la obtención de reglas con mayor generalidad con su función de generación de población inicial. En ella se define un porcentaje de la población inicial que se generará de forma completamente aleatoria. El resto se producirá también de manera aleatoria, pero con un parámetro que indica el número **máximo** de variables que participarán en la regla. Con este esquema, permitimos una mayor generalidad de las reglas generadas.

3.2.4. *Función de evaluación.*

Los pasos a seguir hasta obtener el valor fitness de los individuos y rellenar la población élite son los siguientes:

Primero se buscan todos los individuos no dominados y dominados. Llamaremos *fuerza del individuo* al número de individuos que dicha regla domina. A continuación, se calcula un valor fitness inicial A_i para cada individuo. Este valor A_i es la suma de la fuerza de todos los dominadores del individuo *i*. Por lo que este valor habrá que **minimizarlo** y tiene un valor de cero para los individuos no dominados.

Debido a que este sistema podría fallar cuando hay muchos individuos no dominados, ya que todos tienen el valor máximo de la función, se incluye información adicional sobre la densidad. Esta densidad se calcula con el esquema del vecino más cercano y la siguiente fórmula:

$$D(i) = \frac{1}{\sigma^k + 2}$$

Ecuación 13

Donde σ^k es el valor del k-ésimo vecino más cercano.

El valor de adaptación final es la suma del fitness inicial y el valor de densidad.

$$Fitness(i) = D(i) + A(i)$$

Ecuación 14

Esta función deberemos **minimizarla.**

3.2.5.

Función de truncado

Una vez tenemos el valor fitness de cada individuo y sabemos qué individuos son dominados y no dominados, el siguiente paso consiste en rellenar la población élite con todos los individuos no dominados. Ya que la población élite es de tamaño fijo, es posible que haya más individuos no dominados que el tamaño de la población élite haciendo necesario truncar la población. Para realizar este truncado se toman todos los individuos no dominados, se calcula la distancia entre ellos y, de entre los dos más cercanos, se elimina el que tenga el k-ésimo vecino más cercano con un valor de distancia menor . Este proceso se repite tantas veces como sea necesario hasta que el número de individuos quepa en la población élite. Realizamos este

método para que las soluciones que añadamos a la población élite cubran gran parte del frente de Pareto, intentando por lo tanto aumentar la diversidad de las posibles soluciones.

3.2.6. *Función de rellenado*

Es posible que haya menos individuos no dominados que el tamaño de la población élite. En este caso será necesario añadir individuos dominados. Para ello se ordena la población por el valor de adaptación de cada individuo y se rellena la población élite con los *n* primeros siendo *n* el tamaño de la población élite.

3.2.7. *Operadores genéticos*

Los operadores genéticos serán los siguientes:

- El operador de cruce será un operador de cruce en dos puntos, al igual que en SDIGA.
- El operador de mutación será un operador de mutación sesgado. Este operador es el mismo que para SDIGA y se explica detalladamente en la Sección 3.1.2.5.
- Un operador de selección que consiste en una selección con reemplazo por torneo binario. Esta se realiza únicamente sobre los individuos de la población élite y se seleccionan individuos hasta tener un número de ellos igual a la población inicial. Normalmente el tamaño de la población élite es mucho menor que el de la población inicial, así que tendremos muchos individuos repetidos, por lo que este proceso tiene una fuerte presión selectiva.
- Un operador de reemplazo de la población seleccionada por la población resultante de cruces y mutación basado en el reemplazo directo de los *k* peores individuos de la población, siendo *k* el número de individuos resultantes de cruces y mutaciones. En este caso, no se mira el valor de adaptación de los nuevos individuos.

3.3. Algoritmo NMEEF-SD

NMEEF-SD es un algoritmo genético multiobjetivo que usa elitismo por dominancia de Pareto. Está basado en el algoritmo NSGA II [32] el cual posee un mecanismo rápido de ordenación por dominancia. Para intentar generar mayor diversidad, este algoritmo posee un método de reinicialización de la población basado en cobertura si la población no evoluciona durante una cierta cantidad de evaluaciones.

Actualmente, NMEEF-SD solo trabaja con representacion de reglas de tipo canónico. En [34] se presenta un estudio que muestra que la utilización de reglas DNF para NMEEF-SD no obtiene unos resultados favorables y por lo tanto se descarta la utilización de este tipo de representación.

El funcionamiento del algoritmo es el siguiente: se parte de una población inicial P_t generada de manera sesgada al igual que en MESDIF (Seccion 3.2.3). A continuación, realizamos la selección de una población intermedia Q_t a través de una selección por torneo binario. Los individuos se seleccionan en función del frente al que pertenezcan y de su "*crowding distance"*. Esta población tendrá el mismo tamaño que P_t. Posteriormente a los individuos de Q_t se le aplican los operadores genéticos. Después, se unen P_t y Q_t formando una nueva población R_t. Esta población R_t se ordena según la siguiente forma: primero se encuentran los individuos no dominados, seguidamente los individuos que son dominados por solo un individuo, después los que son dominados por dos y así sucesivamente. Una vez tenemos todos los frentes ordenados según su nivel de dominancia se comprueba si el frente de Pareto evoluciona. Se define como evolucionar si el frente de Pareto actual cubre al menos un nuevo ejemplo que no estaba cubierto por el frente en la iteración anterior.

Una vez comprobamos si ha evolucionado se generará la población que participará en el proceso evolutivo en la próxima iteración. Esta población P_{t+1} se puede obtener de dos formas:

- Si el frente de Pareto no evoluciona durante un 5 % de las evaluaciones totales, P_{t+1} se generará utilizando una función de reinicialización de la población basada en cobertura, que se explicará mas adelante.

- Si el frente de Pareto evoluciona se copian en P_{t+1} los *k* primeros frentes completos hasta que la población se rellene. Si para un frente dado no caben todos los individuos, este frente se ordena en función de la llamada "*crowding distance"* y se copian a P_{t+1} los primeros individuos hasta rellenar P_{t+1}.

  Al final del proceso evolutivo se devuelve el frente de Pareto como solucion. Sin embargo, estos individuos del frente de Pareto deben de cumplir también una condición de confianza (difusa) mínima para que sean devueltos. Esto se realiza para promover la precisión, ya que se ha hecho mucho hincapié en el proceso evolutivo en la diversidad.

3.3.1. *Operadores genéticos*

Los operados genéticos que utiliza NMEEF-SD son un operador de cruce en dos puntos y un operador de mutación sesgada, al igual que MESDIF y SDIGA. Para generar la población inicial se apoya en el mismo operador de generación de población que MESDIF (Seccion 3.2.3) .

3.3.2. *Operador de reinicialización*

Para generar la población que participará en el proceso evolutivo en la siguiente iteración, como hemos explicado anteriormente se examina si el frente de Pareto evoluciona. En caso de que este frente no sea capaz de evolucionar durante un 5 % del número total de evaluaciones, se aplicará el operador de reinicialización basado en cobertura. Este operador copia los individuos no duplicados del frente de Pareto a la población P_{t+1}, y el resto de individuos se generan de forma aleatoria basada en cobertura, cuyo funcionamiento consiste en intentar cubrir un ejemplo de la clase objetivo no cubierto actualmente (si es posible, sino uno aleatorio) con un número máximo de variables que podemos establecer y que, por defecto, es del 50% de variables.

Operador de selección. Crowding distance.

NMEEF-SD utiliza esta medida de densidad para realizar la selección de la población Qt y despúes para rellenar la población P_{t+1} para añadir individuos cuando no caben todos los de un frente dado en la misma.

3.3.3.

Esta medida se basa en la obtención del mayor *cuboide* que encierra a un punto *i* sin que haya más puntos dentro.

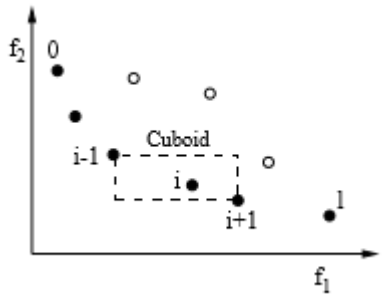


Figura 3.5. Representación del cuboide

Esta distancia se calcula para los puntos que están en el mismo frente (en la imagen representados como puntos negros). Para calcular esta distancia se realizan los siguientes pasos:

- Para cada objetivo, se ordena de menor a mayor el frente en función del valor de ese objetivo.
- Los individuos de los extremos (el mejor y el peor) tienen distancia infinita para que sean siempre seleccionados.
- Para el resto de individuos, la distancia se calcula de la siguiente manera:

$$I[i] = I[i] + (I[i+1].m - I[i-1].m)$$

Ecuación 15

Donde:

- *I* es la distancia actual del individuo.
- *.m* se refiere a cada valor objetivo

Capítulo 4:

Análisis, diseño e implementación de la librería de algoritmos.

4. Análisis, diseño e implementación de la librería de algoritmos.

En este capítulo se decriben las tareas relacionadas con la ingeniería del software, como el análisis de requerimientos y el diseño de nuestra librería de algoritmos. También explicaremos la implementación de la misma, haciendo una breve introducción al lenguaje de programación utilizado para crearla.

4.1. Análisis de requerimientos

A continuación realizaremos un análisis de los requerimientos del sistema. Es una fase muy importante del proyecto ya que definimos lo que se espera que haga el sistema, así como sus posibles restricciones. Todos los requisitos que especifiquemos deberán ser comprobados una vez terminada la implementación para verificar que se cumplen.

4.1.1. *Requerimientos funcionales.*

Los requerimientos funcionales describen qué debe hacer un sistema, es decir, su funcionalidad. Por lo tanto, un requerimiento no debe de especificar el cómo se debe de realizar una cosa. Los requerimientos funcionales de nuestro sistema serán los siguientes:

- Permitir que el usuario pueda modificar los parámetros de cada algoritmo desde un fichero de texto.
- El sistema debe ser capaz de mostrar por pantalla el resultado, así como en diferentes ficheros de salida clasificados según la función que se realice.
- El sistema debe ser capaz de tratar correctamente con ficheros en el formato de datos de KEEL.
- El sistema deberá crear correctamente las particiones difusas que le indique el usuario. Estas serán todas iguales. Además, en la salida, si se utiliza una variable que utilice un conjunto difuso, el sistema deberá indicar claramente cuál es y los límites del mismo.
- El sistema debe de generar mensajes informativos de lo que ocurre a cada momento. En caso de error, deberá informar del error y, si es posible, su posible solución.

Requerimientos no funcionales

Los requerimientos no funcionales describen el grado de cumplimieto de los requerimientos funcionales, normalmente son restricciones que se deben de tener en cuenta de acuerdo a los requisitos del sistema.

4.1.2.

Los requerimientos no funcionales que detectamos para nuestro sistema son los siguientes:

- El sistema deberá ser fácil de usar, por lo que una persona con poca experiencia de uso en R y con cierta experiencia en descubrimiento de subgrupos, o simplemente, conociendo el algoritmo a usar, debe ser capaz de usarlo sin problemas.
- El sistema debe ser capaz de ejecutarse en cualquier sistema operativo donde se tenga R instalado.
- Tiempo de respuesta aceptable, de unos segundos. Aunque este requerimiento depende mucho del tamaño del conjunto de datos.
- El sistema debe ser fácilmente extensible de modo que si queremos mejorar o añadir una utilidad simplemente deberemos modificar mínimamente la librería para incluir el nuevo algoritmo.

4.1.3.

Casos de uso

En este apartado se incluye el diagrama de casos de uso, que permite ejemplificar la interacción del usuario con el sistema en función de los requerimientos definidos anteriormente.

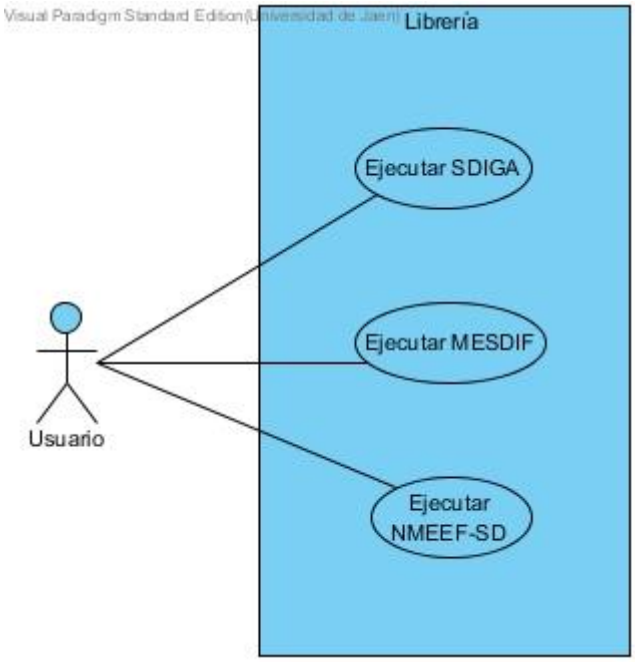


Figura 4.1. Diagrama de casos de uso de la librería

Para explicar los distintos casos de uso, haremos uso de la narrativa, en la que especificaremos con mayor detalle cada una de las interacciones que el usuario tiene con el sistema.

Caso de uso	Ejecutar SDIGA
Actor primario	Usuario.
Condición de entrada	El usuario indica la ruta del fichero de parámetros de entrada.
Condición de salida	El sistema encuentra los subgrupos que mejor se adaptan al conjunto de datos.
Flujo de eventos	1. El usuario ejecuta el algoritmo indicando la ubicación del fichero de parámetros de entrada 2. El sistema devuelve los subgrupos y las medidas de calidad correspondientes
Excepciones	1. a) El fichero de parámetros tiene un formato incorrecto. En este caso se le notifica el error al usuario y cómo solucionarlo. Finaliza el caso de uso. b) La variable objetivo seleccionada no es categórica. Se informa al usuario del error. Finaliza el caso de uso. 2. a) El sistema no encuentra ningún subgrupo que satisfaga las medidas de calidad mínimas. El sistema devuelve el mejor subgrupo obtenido en este caso. Finaliza el caso de uso.

Tabla 4.1. Narrativa del caso de uso "Ejecutar SDIGA"

Caso de uso	Ejecutar MESDIF
Actor primario	Usuario.
Condición de entrada	El usuario indica la ruta del fichero de parámetros de entrada.
Condición de salida	El sistema encuentra los subgrupos que mejor se adaptan al conjunto de datos.
Flujo de eventos	1. El usuario ejecuta el algoritmo indicando la ubicación del fichero de parámetros de entrada. 2. El sistema devuelve los subgrupos y las medidas de calidad correspondientes.
Excepciones	1. a) El fichero de parámetros tiene un formato incorrecto. En este caso se le notifica el error al usuario y cómo solucionarlo. Finaliza el caso de uso. b) La variable objetivo seleccionada no es categórica. Se informa al usuario del error. Finaliza el caso de uso.

Tabla 4.2. Narrativa del caso de uso "Ejecutar MESDIF"

Caso de uso	Ejecutar NMEEF-SD
Actor primario	Usuario.
Condición de entrada	El usuario indica la ruta del fichero de parámetros de entrada.
Condición de salida	El sistema encuentra los subgrupos que mejor se adaptan al conjunto de datos.
Flujo de eventos	1. El usuario ejecuta el algoritmo indicando la ubicación del fichero de parámetros de entrada. 2. El sistema devuelve los subgrupos y las medidas de calidad correspondientes.
Excepciones	1. a) El fichero de parámetros tiene un formato incorrecto. En este caso se le notifica el error al usuario y cómo solucionarlo. Finaliza el caso de uso. b) La variable objetivo seleccionada no es categórica. Se informa al usuario del error. Finaliza el caso de uso. 2. a) El sistema no encuentra ningún subgrupo que satisfaga las medidas de calidad mínimas. El sistema no devuelve subgrupos pero se le notifica al usuario este hecho. Finaliza el caso de uso.

Tabla 4.3. Narrativa del caso de uso "Ejecutar NMEEF-SD"

4.2. Diseño

Una vez especificados los requisitos del sistema deberemos diseñar el sistema para cumplirlos. Esta fase es crítica para la solución correcta del sistema por lo que debe de llevarse a cabo con la máxima atención.

En nuestro caso, implementaremos nuestros algoritmos en el lenguaje de programación R. En la sección 4.3.1 explicaremos con detalle las características del lenguaje.

La implementación de los algoritmos se ha desarrollado utilizando como referencia el código fuente en Java de estos algoritmos disponibles ,en la plataforma de código abierto KEEL [35]. Para la mejor comprensión de los diferentes elementos que forman parte de cada algoritmo, se incluyen a continuación los diagramas de clases y de secuencia de las implementaciones en Java de estos algoritmos.

4.2.1. *Diseño de clases del Sistema*

En este apartado se indica la estructura de los distintos algoritmos, de acuerdo con la implementación en Java disponible en la plataforma KEEL.

4.2.1.1. Diagrama de clases de SDIGA

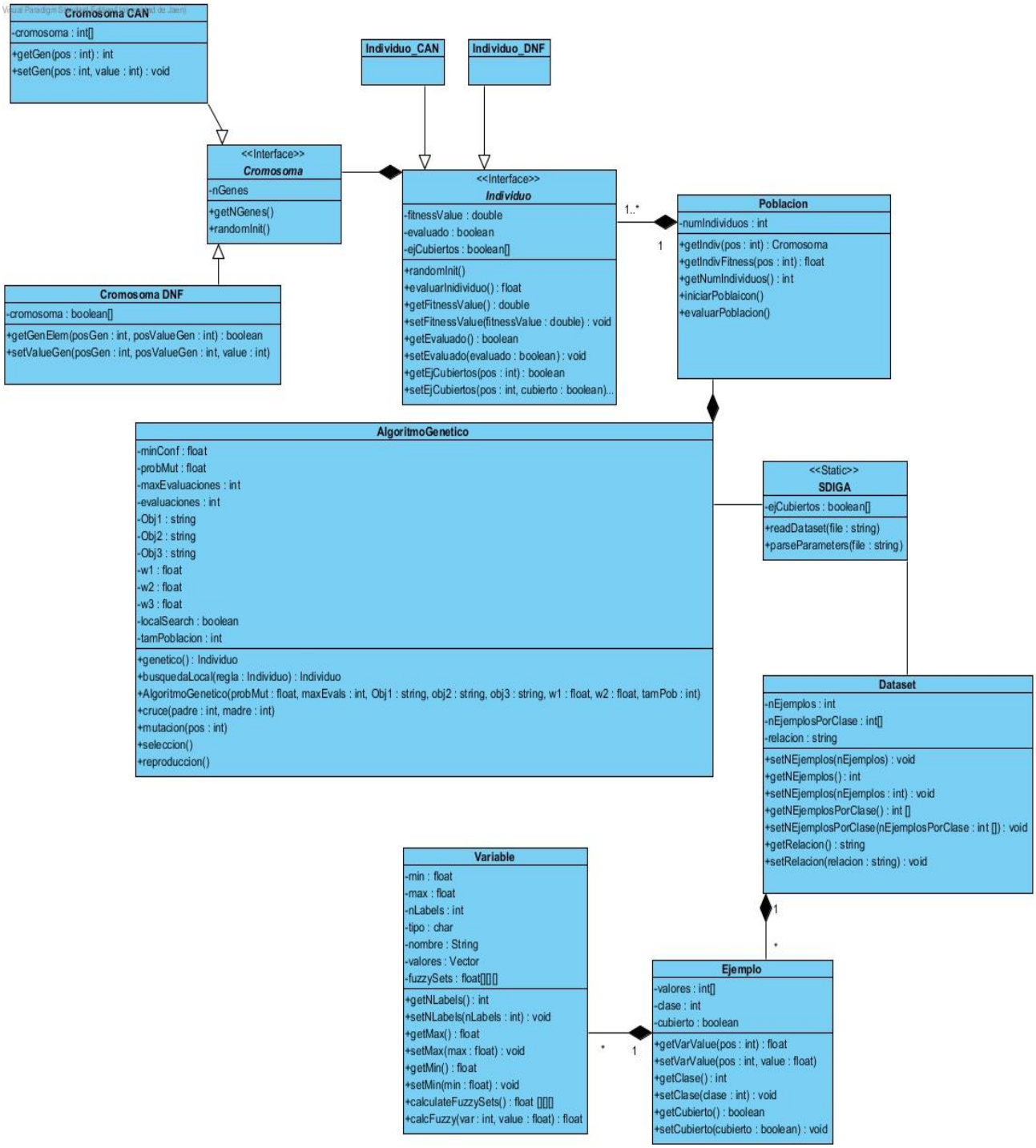


Figura 4.2. Diagrama de clases de SDIGA

4.2.1.2. Diagrama de clases de MESDIF

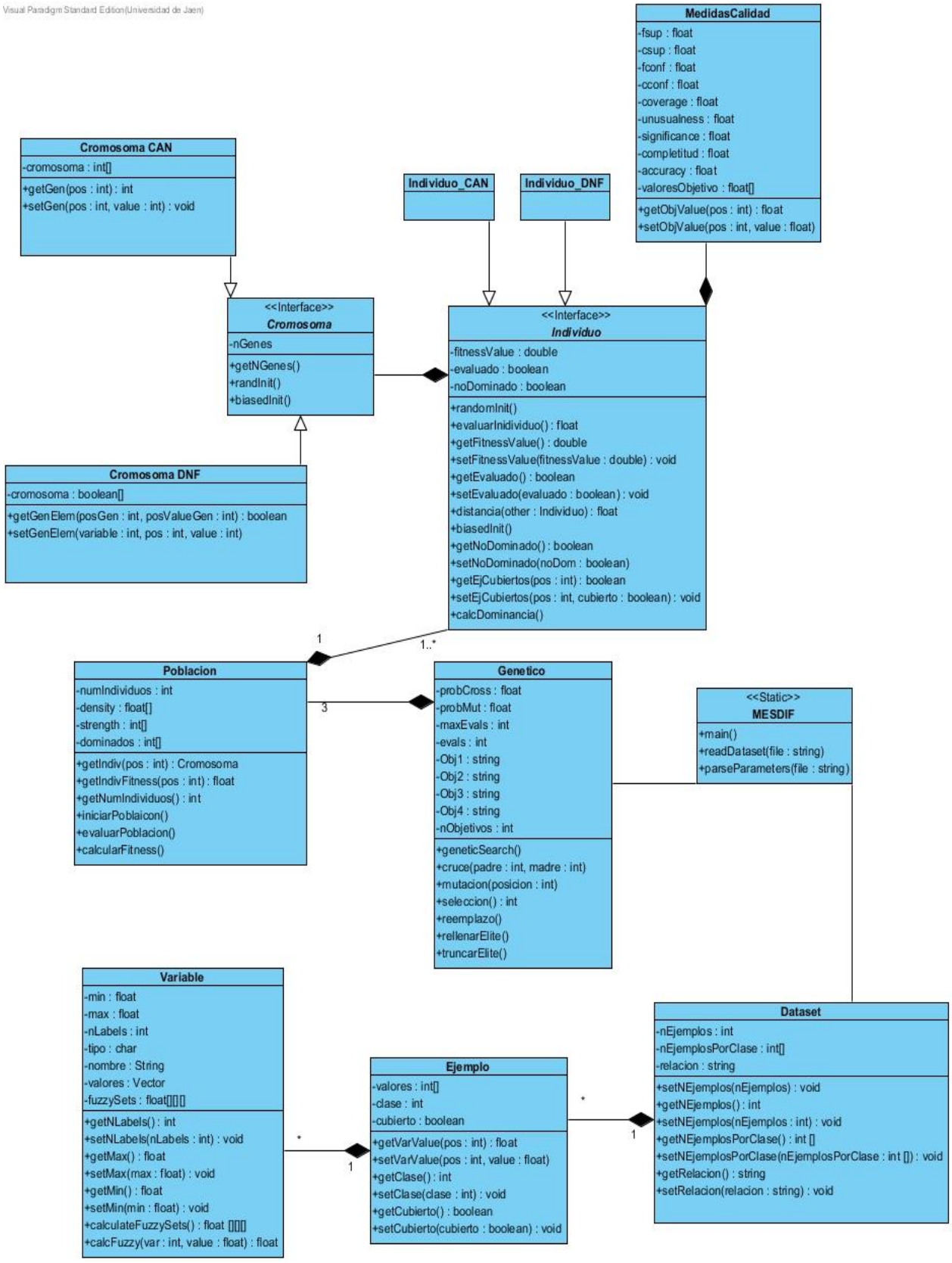


Figura 4.3. Diagrama de clases de MESDIF

4.2.1.3. Diagrama de clases de NMEEF-SD

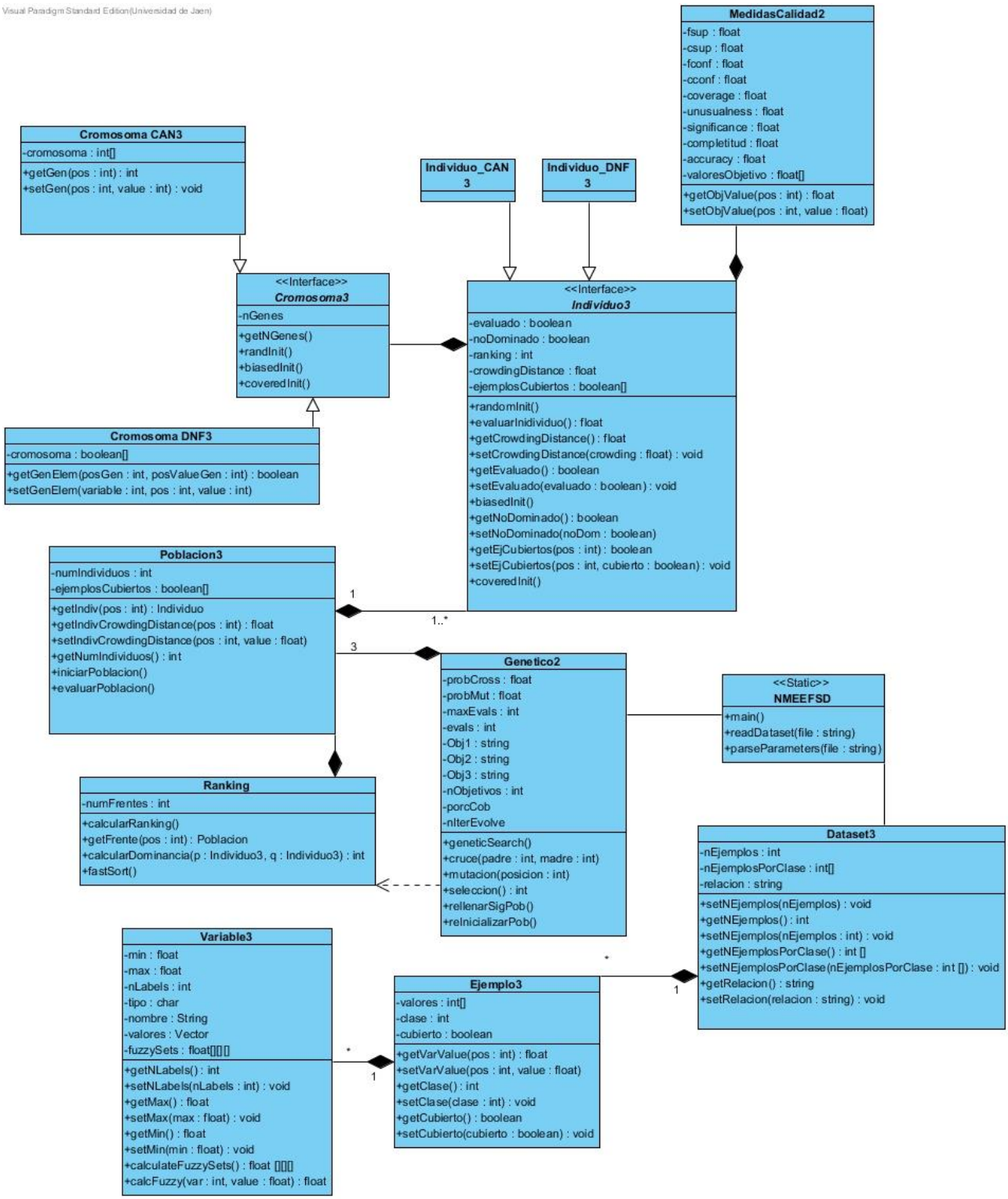


Figura 4.4. Diagrama de clases de NMEEF-SD

Diseño de la secuencia de actividades del Sistema

De igual forma, para facilitar la comprensión de los algoritmos desarrollados, se incluye aquí los diagramas de secuencia correspondientes a los casos de uso relativos a los distintos algoritmos.

4.2.2.

4.2.2.1. Diagrama de secuencia de SDIGA

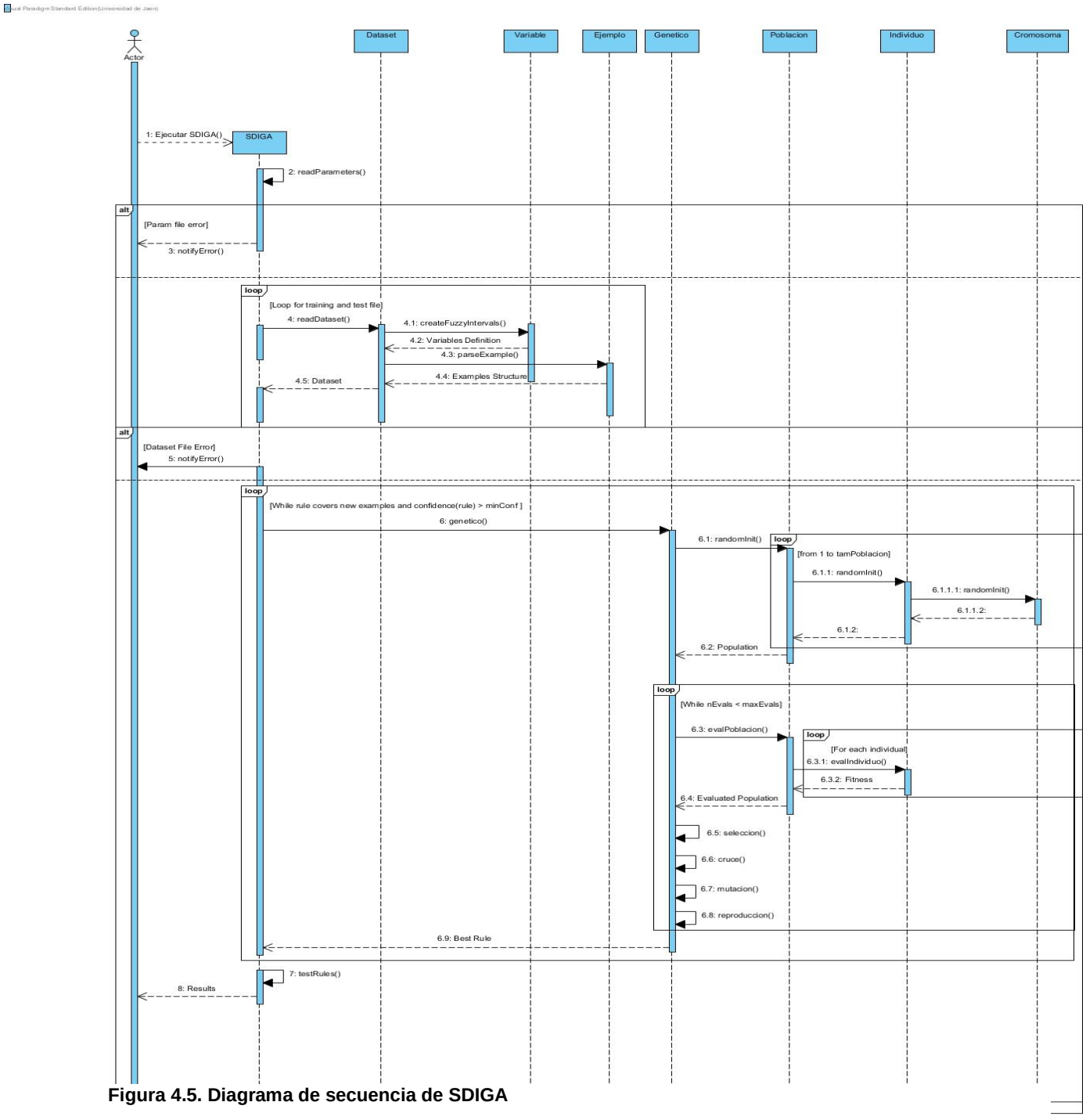


Figura 4.5. Diagrama de secuencia de SDIGA

4.2.2.2. Diagrama de secuencia de MESDIF

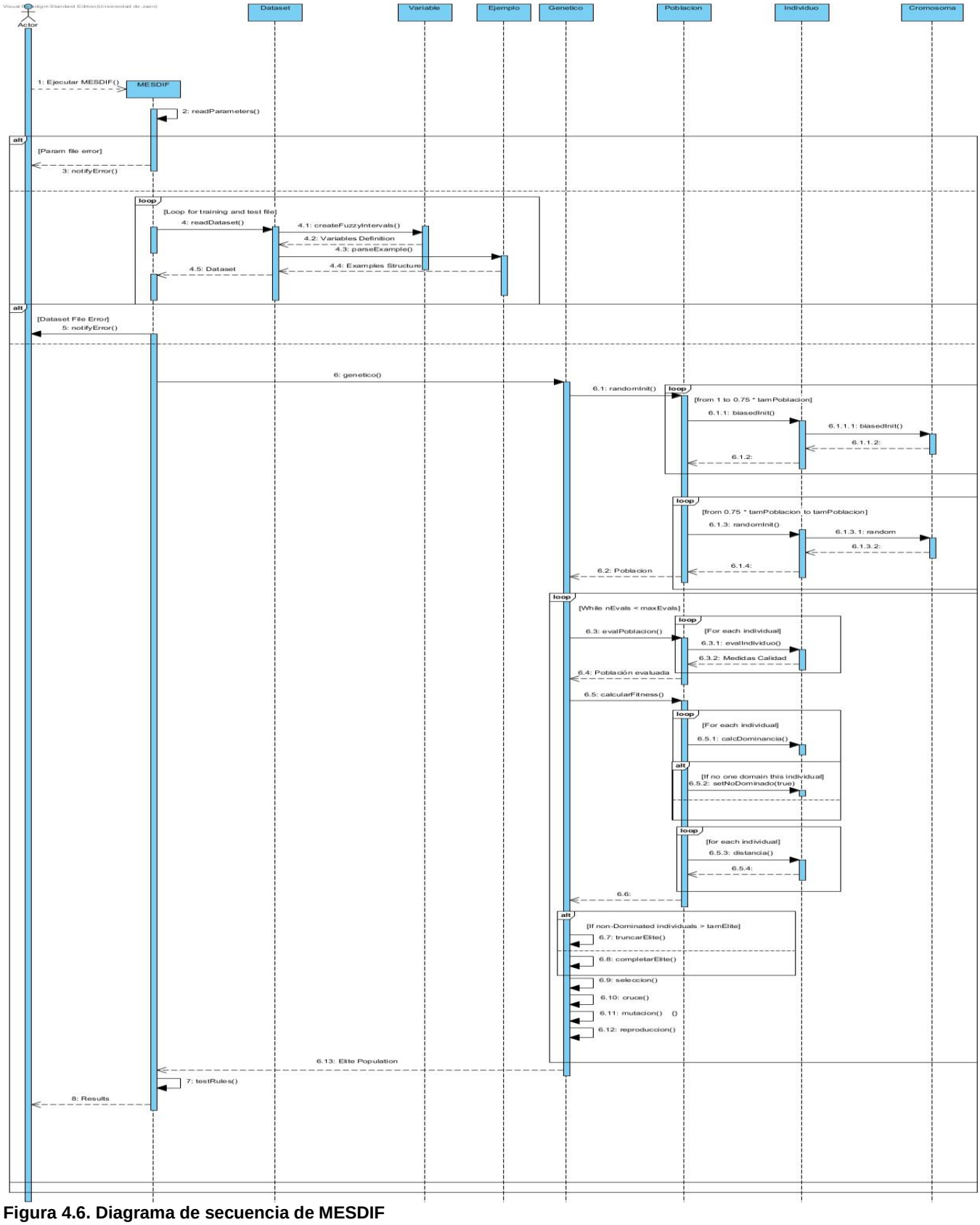


Figura 4.6. Diagrama de secuencia de MESDIF

4.2.2.3. Diagrama de secuencia de NMEEF-SD

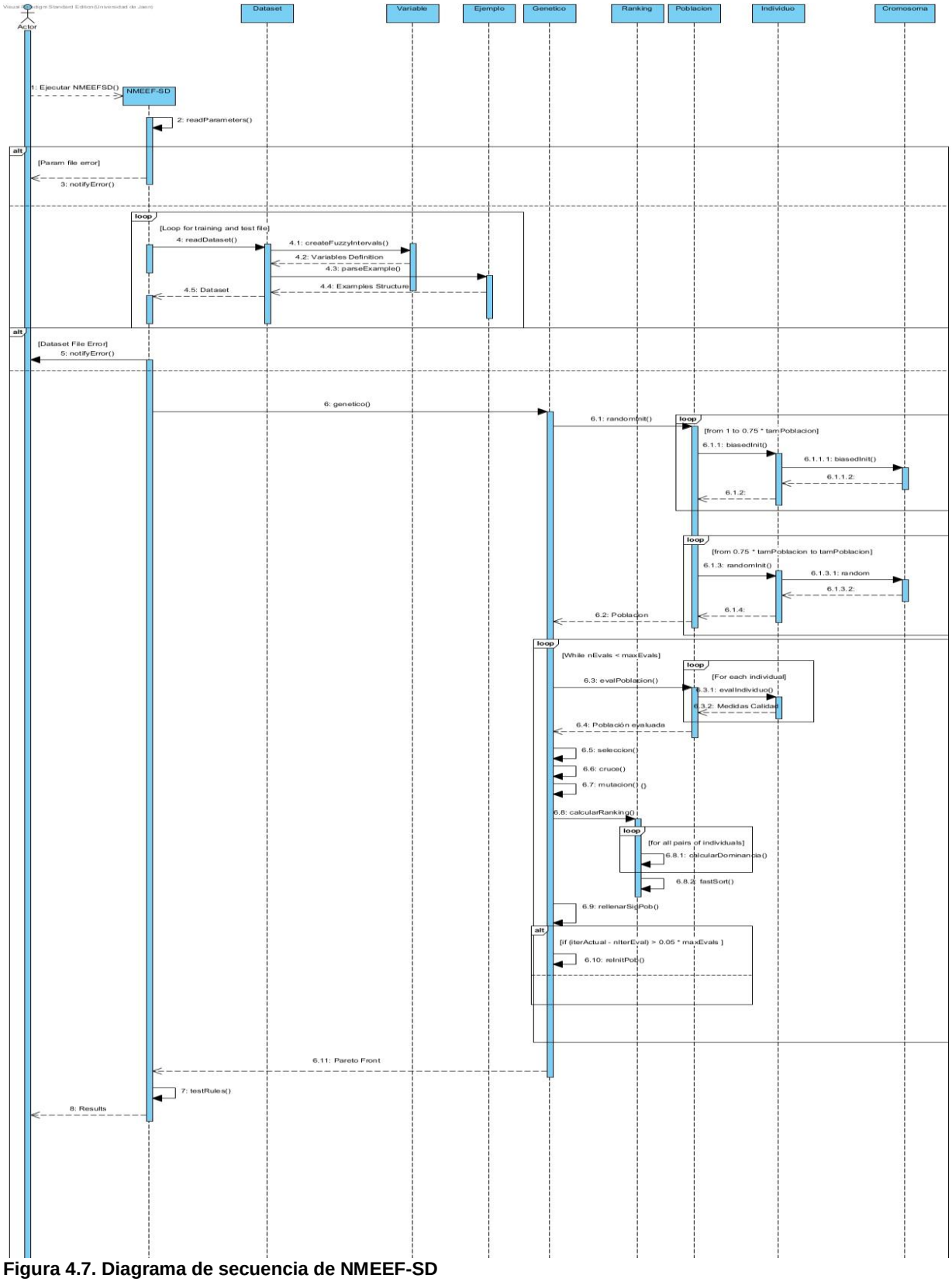


Figura 4.7. Diagrama de secuencia de NMEEF-SD

4.3. Implementación y pruebas

En este apartado se describen las herramientas utilizadas para la implementación de los algoritmos, las características del lenguaje de programación utilizado y el desarrollo de las pruebas necesarias para validar el funcionamiento correcto de los diferentes algoritmos implementados.

4.3.1 *Lenguaje de programación*

El lenguaje de programación que utilizaremos será R [11]. R es un sistema para cálculo estadístico y creación de gráficos que incluye, entre otras cosas, un lenguaje de programación, interfaces de comunicación con otros lenguajes, etc.

En lo referido al lenguaje de programación, R incluye características de programación funcional y de programación orientada a objetos. A pesar de tener un esquema híbrido, el núcleo de R está basado en programación funcional y, por tanto, posee muchas herramientas para poder tratar con funciónes. Podemos, por ejemplo, asignarlas a una variable, almacenarlas en una lista, pasarlas como argumento en otra función, crearlas dentro de otras funciones e incluso, poder devolver en una función otra funcion.

En R, todos los tipos de datos básicos son en realidad objetos, por lo tanto, no se puede acceder directamente a memoria para ver o modificar el dato. El sistema de orientación a objetos de R implementa tres tipos de sistemas de orientación a objetos aparte del sistema base, que difieren en cómo se definen las clases y los métodos de estas clases. Estos sistemas son llamados S3, S4 y Reference Classes (RC). No entraremos en detalles de cómo R implementa estos sistemas, pero comparando el uso de este sistema con las clases básicas de R llegamos a la conclusión de que no son eficientes para nuestro sistema, por lo que basaremos nuestro trabajo en el uso de las clases proporcionadas por el sistema base siempre que sea posible. En la Figura 4.9 vemos el tiempo que tarda un método de las diferentes clases en devolver un dato. Primero se usa el sistema base, luego S3, S4 y por último RC, el resultado es para 100 ejecuciones independientes en tiempo relativo, siendo el valor 1 el mejor tiempo obtenido.

```r
#1
f <- function(x) NULL

#2
s3 <- function(x) UseMethod("s3")
s3.integer <- f

#3
A <- setClass("A", representation(a = "list"))
setGeneric("s4", function(x) standardGeneric("s4"))
setMethod(s4, "A", f)

#4
B <- setRefClass("B", methods = list(rc = f))

a <- A()
b <- B$new()
```

Figura 4.8. Funciones usando los diferentes sistemas de POO de R

```
> microbenchmark::microbenchmark(
+     fun = f(),
+     S3 = s3(1L),
+     S4 = s4(a),
+     RC = b$rc()
+ , unit = "relative")
Unit: relative
 expr min       lq     mean   median       uq       max neval
  fun NaN  1.00000  1.00000  1.00000  1.00000  1.000000   100
   S3 Inf 21.47966 24.14231 23.97645 25.97430  5.687701   100
   S4 Inf 60.93790 62.98149 64.43362 67.43041  9.000268   100
   RC Inf 78.91863 81.29163 82.91435 86.41113 12.062969   100
```

Figura 4.9. Comparativa de tiempo entre sistemas de POO de R

R, al igual que el resto de lenguajes de programación, es capaz de ampliar sus posibilidades a base de librerías. Estas librerías en R son llamadas paquetes, los cuales se pueden instalar e integrar fácilmente en R. Actualmente existen gran cantidad de paquetes, muchos de los cuales están implementados en C/C++ ya que proporcionan mejor eficiencia en procesos computacionalmente costosos.

En cuanto al tratamiento de datos, R es especialmente potente en el manejo de vectores y matrices de datos. R proporciona una infinidad de operadores para realizar muchas de las operaciones más comunes con matrices y vectores. Además, los operadores aritméticos y lógicos clásicos (suma, resta, and y or, entre otros) están "vectorizados". Esto quiere decir que, por ejemplo, la suma de vectores de igual tamaño (o uno múltiplo del otro) se puede realizar elemento a elemento simplemente usando *vector1 + vector2*. Por otra parte, dicha implementación (la cual incluye un bucle) está hecha en C, por lo que es muy rápida. Para comprobar la diferencia entre el uso de operaciones "vectorizadas", bucles clásicos y funciones *funcionales*[2] hemos realizado una pequeña comprobación de tiempos entre estas funciones.

```
sumaVectorizada <- function(vector){
  vector <- vector + 1
  vector
}


sumaFor <- function(vector){
  for(i in seq_len(length(vector))){
    vector[i] <- vector[i] + 1
  }
  vector
}

sumaLapply <- function(vector){
  vector <- lapply(X = vector, FUN = '+', 1)
  vector
}
```

Figura 4.10. Funciones para pruebas de eficiencia de R

```
> values <- numeric(1e6)
> microbenchmark::microbenchmark(sumaFor(values), sumaVectorizada(values), sumaLapply(values), unit = "relative" )
Unit: relative
                 expr      min       lq     mean   median       uq      max neval
        sumaFor(values) 726.4134 703.3177 323.8366 682.3131 436.1035 22.139637   100
 sumaVectorizada(values)   1.0000   1.0000   1.0000   1.0000   1.0000  1.000000   100
     sumaLapply(values) 225.4445 244.1662 111.5141 237.7001 152.1242  8.110245   100
> |
```

Figura 4.11. Comparativa de tiempos de ejecución de funciones de R

Las funciones representadas en la Figura 4.10 aumentan en 1 el valor de todos los elementos de un vector y luego lo devuelve. En este caso, el vector tiene un millón de elementos. El resultado corresponde a 100 ejecuciones independientes en tiempo relativo, siendo el valor 1 el mejor resultado. Como se observa en la Figura 4.11, el uso de operaciones vectorizadas es, de media, 323 veces más rapido que el uso de

[2] Una función *funcional* es una función que usa a otra función pasada como argumento.

un bucle for y 111 veces más rápido que el uso de un funcional como lapply. Por lo tanto, intentaremos utilizar siempre que podamos la vectorización, algo que no es sencillo de aplicar y que no siempre se puede realizar, teniendo que utilizar en estos casos, sin mas remedio, los lentos bucles for.

Entorno de desarrollo

Para desarrollar los algoritmos en el lenguaje de programación utilizaremos el entorno de desarrollo RStudio[3]. Este nos proporciona una consola de R, un editor de texto que nos permite resaltado de funciones y palabras clave, fácil integración de funciones en el entorno de R y *debugging* sencillo de las funciones añadidas, así como ventanas para visualizar las graficas que generemos en R, la ayuda de las diversas funciones, etc. También nos permite crear y compilar paquetes de manera sencilla, integración sencilla de herramientas de control de versiones, crear aplicaciones web con *Shiny* así como lanzarlas de manera sencilla, etc. Es, sin duda, uno de los mejores entornos de desarrollo para R hasta el momento.

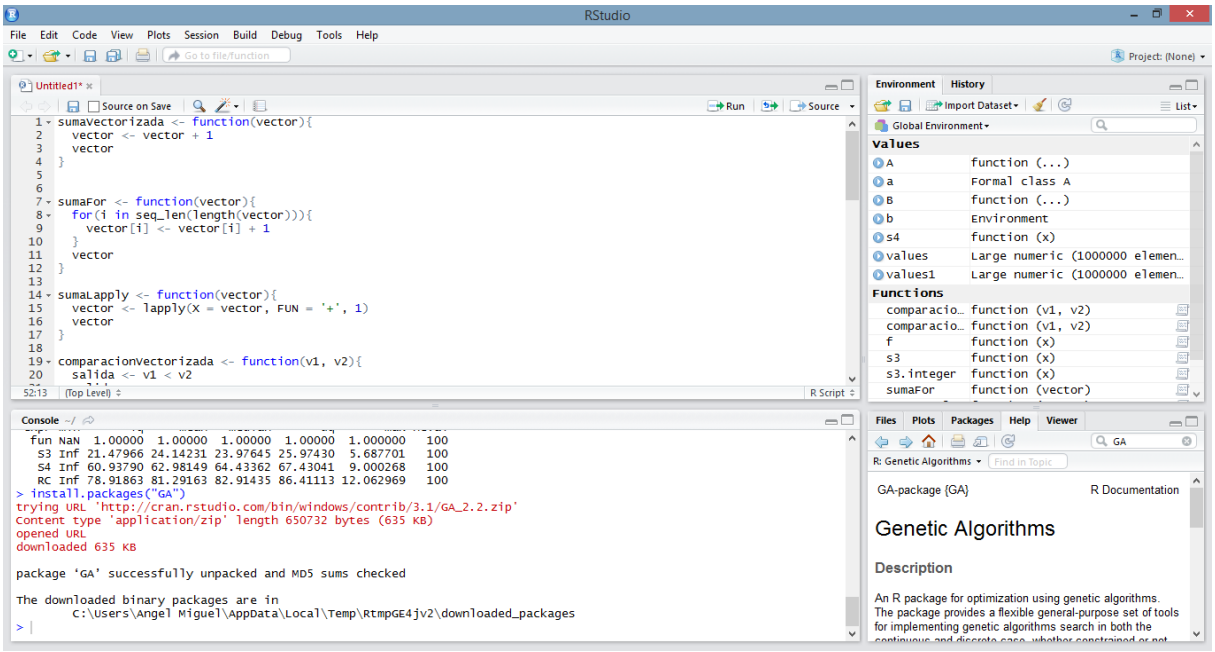


Figura 4.12. RStudio

[3] http://www.rstudio.com

4.3.3. *Pruebas*

Para el desarrollo de las pruebas se ha utilizado el enfoque TDD (Test-driven development). En este enfoque se definen los casos de prueba que deben superar cada uno de los elementos antes incluso de implementarlos. De esta forma, el código se desarrolla con menos errores y permite enfoques de validación automática. Para la comprobación de la correcta funcionalidad se ha utilizado como apoyo los algoritmos implementados en Java y se ha verificado que, a igualdad de parámetros y datos, las salidas son idénticas.

A continuación se recogen los resultados obtenidos con una serie de bases de datos de prueba. En la sección 6.1.1 se comentan las características de estas bases de datos.

a) Dataset *Iris*

	Reglas CAN			Reglas DNF		
	SDIGA - JAVA	**SDIGA - R**	**Diferencia**	**SDIGA - JAVA**	**SDIGA - R**	**Diferencia**
nº reglas	3	3,2	0,2	3	3	0
nº variables	2,36	2,34	-0,02	2,56	2,36	-0,2
coverage	0,3844400	0,3727732	-1%	0,4665786	0,4599802	-1%
significance	6,3610758	6,6171772	0,2561014	7,6347708	7,9696152	0,3348444
unusualness	0,1582962	0,1632962	1%	0,1733334	0,1777776	0%
accuracy	0,7108812	0,7266364	2%	0,6773684	0,6922928	1%
csupport	0,8599340	0,8799340	2%	0,9866600	0,9933334	1%
fsupport	0,2518582	0,2539452	0%	0,2543514	0,2544822	0%
fconfidence	0,7469036	0,7614616	1%	0,6709041	0,6814546	1%
cconfidence	0,8430548	0,8567928	1%	0,7674772	0,7879048	2%

Tabla 4.4. Resultados de prueba con el dataset Iris para SDIGA

	Reglas CAN			Reglas DNF		
	MESDIF - JAVA	MESDIF - R	Diferencia	MESDIF - JAVA	MESDIF - R	Diferencia
nº reglas	9	9	0	6,75	7	0,25
nº variables	2,583333425	2,53333352	-0,049999905	2,54761850	2,7714288	0,2238103
coverage	0,385833225	0,37037020	-2%	0,44484125	0,3771428	-7%
significance	5,367613250	5,09159980	-0,27601345	5,38341500	4,5365740	-0,846841
unusualness	0,134259500	0,12254340	-1%	0,13664025	0,1104764	-3%
accuracy	0,643059000	0,61823740	-2%	0,62820550	0,5837540	-4%
csupport	0,983333250	0,99333340	1%	1	1	0%
fsupport	0,236420000	0,22382140	-1%	0,23683322	0,2153562	-2%
fconfidence	0,689455500	0,68194260	-1%	0,63938500	0,6090318	-3%
cconfidence	0,749173750	0,69521660	-5%	0,74097325	0,6554646	-9%

Tabla 4.5. Resultados de prueba con el dataset Iris para MESDIF

	Reglas CAN		
	NMEEFSD - JAVA	NMEEFSD - R	Diferencia
nº reglas	7,8	7,8	0
nº variables	2,8304114	2,8304114	0
coverage	0,2743648	0,2743648	0%
significance	6,4085516	6,4085518	2E-07
unusualness	0,1575242	0,1575794	0%
accuracy	0,7323626	0,7323626	0%
csupport	0,8866666	0,8866666	0%
fsupport	0,2276962	0,2276962	0%
fconfidence	0,8057544	0,8057544	0%
cconfidence	0,9039042	0,9039042	0

Tabla 4.6. Resultados de prueba con el dataset Iris para NMEEF-SD

b) Dataset *monk-2*

	Reglas CAN			Reglas DNF		
	SDIGA - JAVA	SDIGA - R	Diferencia	SDIGA - JAVA	SDIGA - R	Diferencia
nº reglas	4	4	0	2	2	0
nº variables	2	2	0	2,4	2	-0,4
coverage	0,3124504	0,3124504	0%	0,8332932	0,8332932	0%
significance	7,7505610	7,7505610	0	2,6710372	2,6710368	-4,4E-07
unusualness	0,1070836	0,1466512	4%	0,0741230	0,0832486	1%
accuracy	0,8290316	0,8290316	0%	0,6078462	0,6078462	0%
csupport	0,9721732	0,9721732	0%	0,9721732	0,9721730	0%
fsupport	0,2684910	0,2684910	0%	0,4860866	0,4840864	0%
fconfidence	0,8060076	0,8060076	0%	0,6117832	0,6124632	0%
cconfidence	0,8548608	0,8548608	0%	0,6117832	0,6117830	0%

Tabla 4.7. Resultados de prueba con el dataset monk-2 para SDIGA

	MESDIF - JAVA	MESDIF - R	Diferencia	MESDIF - JAVA	MESDIF - R	Diferencia
nº reglas	6	6	0	6	6	0
nº variables	2,6333332	2,4333332	-0,2	2,7333336	2,5333334	-0,2000002
coverage	0,2458690	0,2484408	0%	0,6229486	0,6438608	2%
significance	5,9812958	6,2749528	0,293657	7,1127288	7,6717842	0,5590554
unusualness	0,0632538	0,1238472	6%	0,0930468	0,1105074	2%
accuracy	0,7721298	0,8057450	3%	0,7214230	0,7239018	0%
csupport	0,7590350	0,8008552	4%	1	1	0%
fsupport	0,1849672	0,1955728	1%	0,3998648	0,4144362	1%
fconfidence	0,7624828	0,7994642	4%	0,6897064	0,6957274	1%
cconfidence	0,8108190	0,8526772	4%	0,7363522	0,7379186	0%

Tabla 4.8. Resultados de prueba con el dataset monk-2 para MESDIF

	Reglas CAN		
	NMEEFSD - JAVA	NMEEFSD - R	Diferencia
nº reglas	5,4	5,4	0
nº variables	2,2533334	2,2533334	0
coverage	0,3116680	0,3116680	0%
significance	7,0386040	7,0386042	2E-07
unusualness	0,0972352	0,1513266	5%
accuracy	0,8263430	0,8263430	0%
csupport	0,9282814	0,9282812	0%
fsupport	0,2231780	0,2231780	0%
fconfidence	0,8225556	0,8225556	0%
cconfidence	0,8586914	0,8586914	0

Tabla 4.9. Resultados de prueba con el dataset monk-2 para NMEEF-SD

Como se puede apreciar, existen ligeras diferencias entre ambas implementaciones. Esto se debe principalmente a dos factores:

- La forma de generar un número aleatorio para seleccionar individuos, elegir puntos de cruce, valores de mutación, etc. difiere de un lenguaje a otro. Los algoritmos evolutivos son algoritmos estocásticos y su resultado depende en gran medida de la secuencia de números aleatorios generada.
- La realización de los operadores genéticos en R está principalmente vectorizada por eficiencia. Esto implica que, por ejemplo, a la hora de realizar la selección, se seleccionen todas las parejas de individuos de una sola vez y a continuación, de igual manera, se crucen y muten. Este proceso, es iterativo en la implementación de Java, es decir, se seleccionan dos individuos, se cruzan y mutan y el proceso vuelve a iterar. Por lo tanto, al diferir en este proceso, no se seleccionan los mismos individuos en el cruce, no se mutan las mismas variables, etc. por lo que el resultado variará ligeramente.

Capítulo 5:

Análisis, diseño e implementación de la interfaz web.

5. Análisis, diseño e implementación de la interfaz web

Al igual que en el capítulo 4 de esta memoria, se describe aquí todo el proceso de análisis, diseño e implementación de la interfaz web que hará posible el uso de los algoritmos implementado en la librería de R por cualquier persona que tenga un navegador web.

5.1. Análisis de requerimientos

Procederemos a continuación a explicar los requerimientos funcionales y no funcionales que debe cumplir la interfaz web.

5.1.1. *Requerimientos funcionales*

Los requerimientos que debe cumplir nuestra interfaz serán los siguientes:

- Se debe permitir al usuario la posibilidad de cargar un fichero de datos de entrenamiento y otro de test en el formato de datos de KEEL.
- Se debe permitir, en función del conjunto de datos escogido, la selección de una variable objetivo, así como un valor de la misma o la búsqueda de subgrupos para todas las clases.
- Se debe permitir la selección de los diferentes algoritmos que conforman la librería, así como la modificación sencilla de los parámetros que utiliza.
- Se debe permitir la visualización de diferentes propiedades del conjunto de entrenamiento o de test o de un subconjunto de estos en función de la selección del usuario. Las propiedades deberán visualizarse de una manera clara y útil para el usuario.
- La interfaz deberá ejecutar correctamente el algoritmo seleccionado con los parámetros especificados manualmente por el usuario. En caso de error en algún parámetro, se debe de indicar claramente.
- La interfaz deberá mostrar adecuadamente los resultados devueltos por el algoritmo.

Requerimientos no funcionales

Los requerimientos no funcionales de la interfaz web serán los siguientes:

- El tamaño de un fichero de datos a subir en la interfaz web no debe exceder de un tamaño de 10 MB.

5.1.2.

- Las estadísticas deberán reflejar la distribución de ejemplos que tiene un valor concreto. Para intentar reflejar mas claramente el resultado, se deberán usar gráficas adecuadas (como diagramas de barras) al tipo de datos aparte de los valores numéricos estadísticos (que pueden ser media, mediana, desviación típica, etc.).
- Los parámetros que no tengan un valor máximo (numero de evaluaciones, por ejemplo) deberán tener un tope superior para evitar excesivo tiempo de ejecución. Por ejemplo, en el número de evaluaciones deberá ser de 15000 como máximo.
- La interfaz mostrará los resultados en diferentes secciones diferenciadas según el tipo de información que den (reglas generadas, información de la ejecución realizada, medidas de calidad, etc.)
- La interfaz debe informar al usuario, con cualquier tipo de mensaje de lo que está ocurriendo. Por ejemplo, durante la ejecución del algoritmo debe mostrar algún elemento que indique el estado de ejecución.

Casos de uso

A continuación mostraremos el diagrama de casos de uso de la interfaz web, en donde se modelan las posibles interacciones del usuario con el sistema

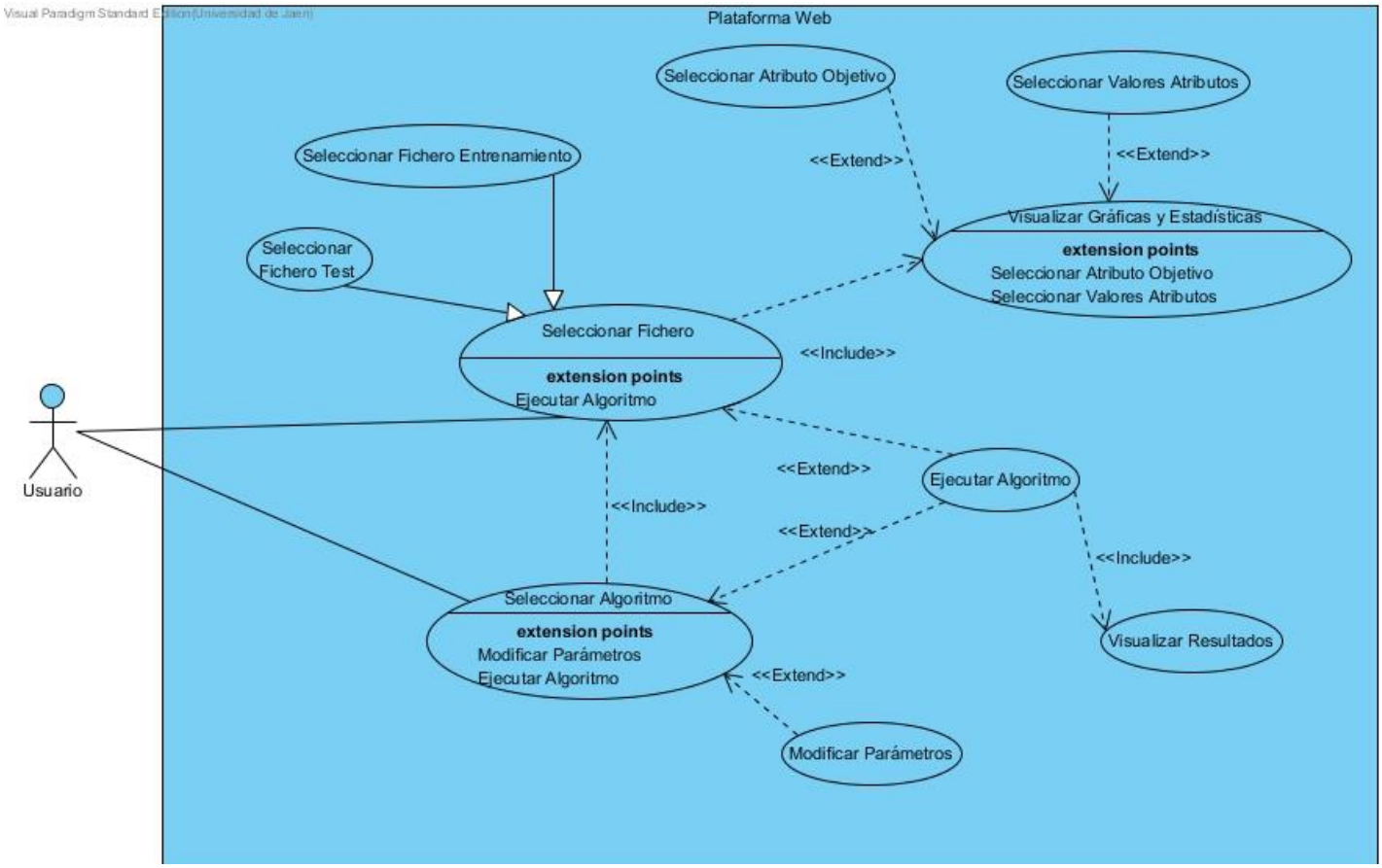


Figura 5.1. Diagrama de casos de uso de la interfaz web.

Aquí especificaremos con mayor detalle los casos de uso más relevantes del sistema.

a) Caso de uso "Seleccionar fichero"

Caso de uso	Seleccionar Fichero
Actor primario	Usuario.
Condición de entrada	El usuario indica la ruta del fichero de entrenamiento/test.
Condición de salida	El sistema muestra una gráfica con información sobre el dataset.
Flujo de eventos	1. El usuario selecciona la ruta del fichero 2. Se muestra al usuario por pantalla la gráfica con datos del dataset.
Excepciones	1. a) El fichero de parámetros tiene un formato incorrecto. En este caso se le notifica el error al usuario y cómo solucionarlo. Finaliza el caso de uso. .

Tabla 5.1. Narrativa del caso de uso "Seleccionar Fichero"

b) Caso de uso "Seleccionar algoritmo"

Caso de uso	Seleccionar Algoritmo
Actor primario	Usuario.
Condición de entrada	El usuario indica el algoritmo que va a utilizar.
Condición de salida	El sistema muestra al usuario los parámetros que el algoritmo necesita.
Flujo de eventos	1. El usuario selecciona el algoritmo a usar. 2. Se muestra al usuario el conjunto de parámetros necesarios para la ejecución del algoritmo seleccionado. 3. El usuario necesita seleccionar a continuación el conjunto de datos de entrenamiento y test. Ir al caso de uso "Seleccionar Fichero".
Excepciones	Ninguna.

Tabla 5.2. Narrativa del caso de uso "Seleccionar Algoritmo"

c) Caso de uso “Visualizar gráficas y estadísticas”

Caso de uso	Seleccionar Algoritmo
Actor primario	Usuario.
Condición de entrada	El usuario tiene desplegada una gráfica inicial.
Condición de salida	El sistema muestra al usuario una gráfica modificada en función de lo que quiera el usuario.
Flujo de eventos	1. a) El usuario elige entre visualizar el fichero de entrenamiento o el de test. 1. b) El usuario selecciona, de entre los valores de la variable objetivo seleccionada, los valores que quiere visualizar. 1. c) El usuario selecciona el tipo de gráfica que desea visualizar.
Excepciones	1. a) No ha sido seleccionado aún el fichero de entrenamiento o de test. No se muestra nada 1. b) El usuario ha seleccionado una variable numérica. El usuario no puede seleccionar valores. 2. c) El usuario selecciona una variable de tipo numérico. En este caso la única gráfica posible es un histograma.

Tabla 5.3. Narrativa del caso de uso "Visualizar gráficas y estadísticas"

d) Caso de uso “Ejecutar algoritmo”

Este caso de uso es el caso de uso de nuestra librería. Por lo que no repetiremos la narrativa.

5.2. Diseño

En esta sección realizaremos el diseño de nuestra interfaz. En este caso utilizaremos una serie de técnicas de diseño orientadas al usuario, ya que nuestro objetivo es hacer que la interfaz sea fácil de utilizar por una gran cantidad de personas (cuantas más, mejor).

5.2.1 *Personas*

La técnica “Personas” nos permite especificar arquetipos de usuarios. Estos se especifican no como un ente abstracto, como por ejemplo “Perfil cliente comprador”, sino como una persona en concreto, con nombre, foto, historia, capacidades, etc. La principal ventaja que nos ofrece este método es que nos permite generar una mejor empatía con el arquetipo de usuario que definamos. Pudiendo así tratar sus dificultades y/o deficiencias en el uso de la interfaz de una manera mas sencilla e intuitiva por parte del diseñador. A continuación definiremos dichas personas.

5.2.1.1. Persona 1: Carlos Alonso Martínez

Objetivos Clave:

Trabajar en sus investigaciones ya que es profesor de universidad en el área de biología. Además le gusta practicar deporte y sale a correr todos los días. Le gusta mucho estar informado de la actualidad y la lectura de libros de ciencia ficción y aventuras.

Figura 5.2. Carlos Alonso Martínez

Habilidades con el ordenador

No es gran fan de los ordenadores, sin embargo es un fanático del *smartphone* y lo utiliza muy a menudo para ver Twitter, tener un seguimiento de su actividad deportiva, etc. El ordenador lo utilza principalmente para sus investigaciones, sabe usar la suite de Office bastante bien y también se defiende en el uso de R y MatLab, aunque a veces les desespera debido a que la programación no es su fuerte.

Un día en su vida.

Carlos se levanta temprano todas las mañana para salir un rato a correr. Cuando termina se dirige a la universidad para trabajar en sus investigaciones. Después de impartir un par de clases continúa trabajando en su despacho en sus investigaciones las cuales últimamente están siendo bastante buenas. Sin embargo, Carlos se queja del tiempo empleado en realizar las investigaciones debido a que la mayoría de métodos de minería de datos que utilizan deben realizarlos a mano y es una tarea ardua. Echa en falta una herramienta que permita hacer todos estos métodos automáticamente.

5.2.1.2. Persona 2: Miguel Álvarez del Castillo

Objetivos clave

Miguel pretende acabar sus estudios de Informática pronto. Actualmente es su tercer año y ya se interesa mucho por la inteligencia artificial en general. Debido a su gran interés, está buscando formas y sitios que sean capaces de enseñarle.

Figura 5.3. Miguel Álvarez del Castillo

Habilidades con el ordenador

Es un experto en el ordenador, maneja varios lenguajes de programación perfectamente, se maneja muy bien en todos los sistemas operativos y maneja bien diferentes herramientas ofimáticas así como suites matemáticas y estadísticas.

Un día en su vida

Miguel se dirige a clase como todos los días. Pregunta muchas dudas al profesor y este se las resuelve. Después de clase, dedica un par de horas más durante la tarde después de realizar sus trabajos de la universidad a ampliar sus conocimientos y por ello busca en Internet información. La información que encuentra en Internet no le proporciona herramientas sencillas para el aprendizaje de los métodos que lee. Sería necesario para Miguel una herramienta que le proporcione con sencillez una visión general de lo que realiza cada método a modo de ejemplo de lo que lee en artículos y otros materiales.

5.2.2. *Escenarios*

Esta técnica recrea una posible situación de uso de nuestro sistema y tiene un gran potencial si la utilizamos en conjunción con la técnica de personas. Ya que podemos identificar fácilmente posibles problemas que ocurren y así darles solución.

5.2.2.1. Escenario 1

Carlos acaba de llegar a su despacho de la Universidad después de haber salido a correr, hoy pretende realizar unos gráficos que le parecen interesantes para incluir en su nuevo artículo. Carlos sabe que nuestra aplicación es capaz de realizar gráficos, por lo que la abre, carga su conjunto de datos y se le muestra la gráfica. Sin embargo,

no encuentra la manera de cómo guardar la imagen y recurre a la aplicación “recortes” de Windows.

5.2.2.2. Escenario 2

Carlos está intentando obtener una serie de resultados para sus investigaciones, para ello necesita guardar los resultados obtenidos en los ficheros que se generan. Como no le gusta mucho escribir en la consola de R, ya que le parece muy tedioso, opta por el uso de la interfaz web para realizar su experimentación. Se lleva una grata sorpresa cuando los resultados obtenidos son grandísimos y tiene que realizar mucho *scroll* con el ratón para visualizar todos los resultados

5.2.2.3. Escenario 3

Miguel está buscando por Internet cosas de su interés cuando encuentra un enlace hacia nuestra plataforma web. Entonces empieza a investigar la página y el primer problema que se encuentra es que no sabe qué es KEEL, por lo que empieza a buscar información sobre él. Una vez aprende, sigue investigando la página. Al añadir un conjunto de datos éste le muestra la gráfica pero no valores estadísticos númericos, lo cual le sorprende y le hace pensar la baja calidad del sitio web.

5.2.3. *Storyboard*

El storyboard es un “prototipo” de la aplicación en el que se define de manera gráfica un caso de uso. En él se muestran todas las posibles interacciones del usuario con la interfaz así como el resultado de como se verán. La principal ventaja de recurrir a este método es que podemos diseñar la interfaz de una forma en la que la lógica de las acciones que lleva a cabo el usuario sea realizada de una forma natural, es decir, sin la necesidad de buscar demasiado en la pantalla dónde se encuentra el siguiente paso a realizar.

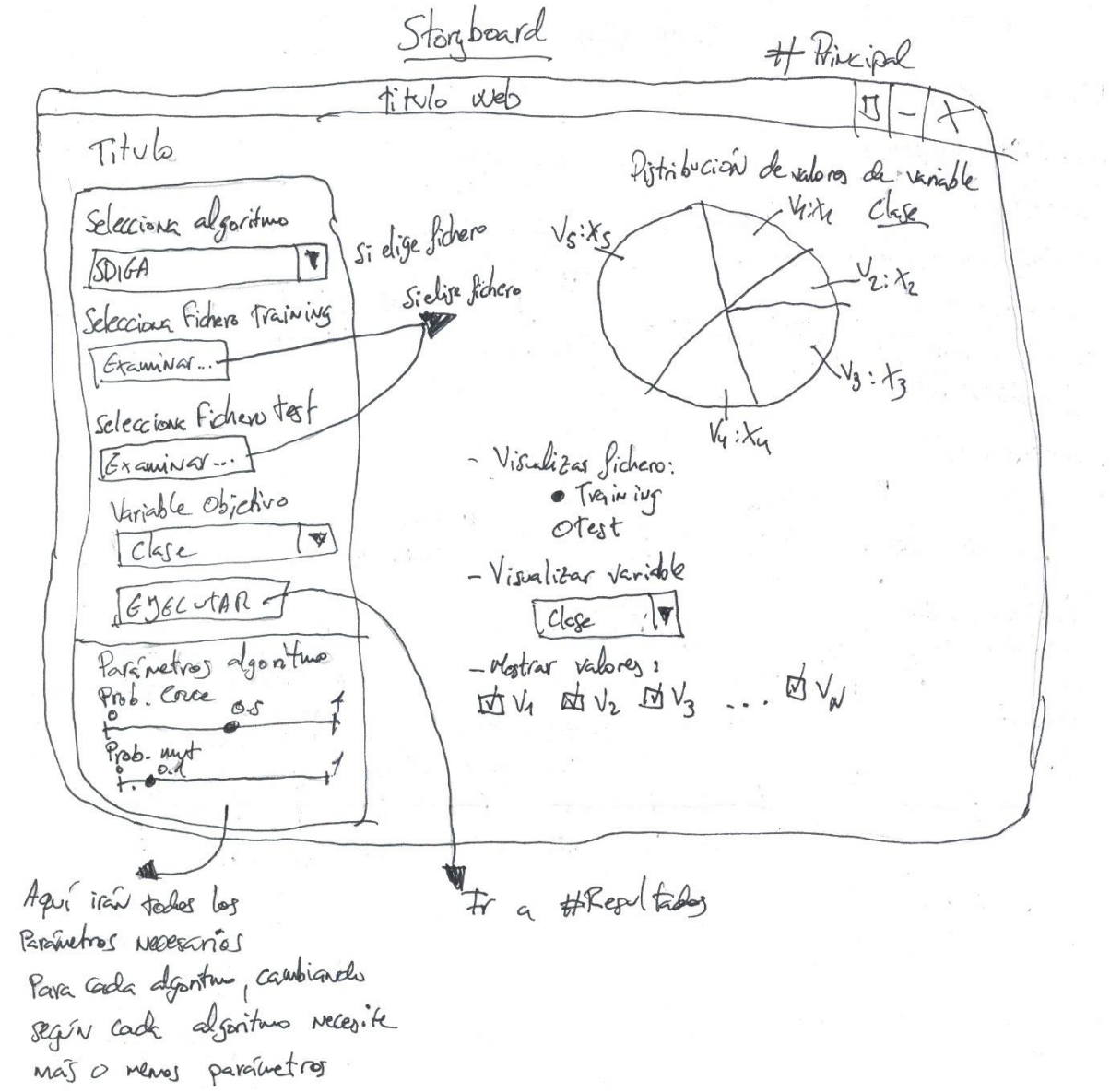


Figura 5.4. Storyboard, pantalla #Principal

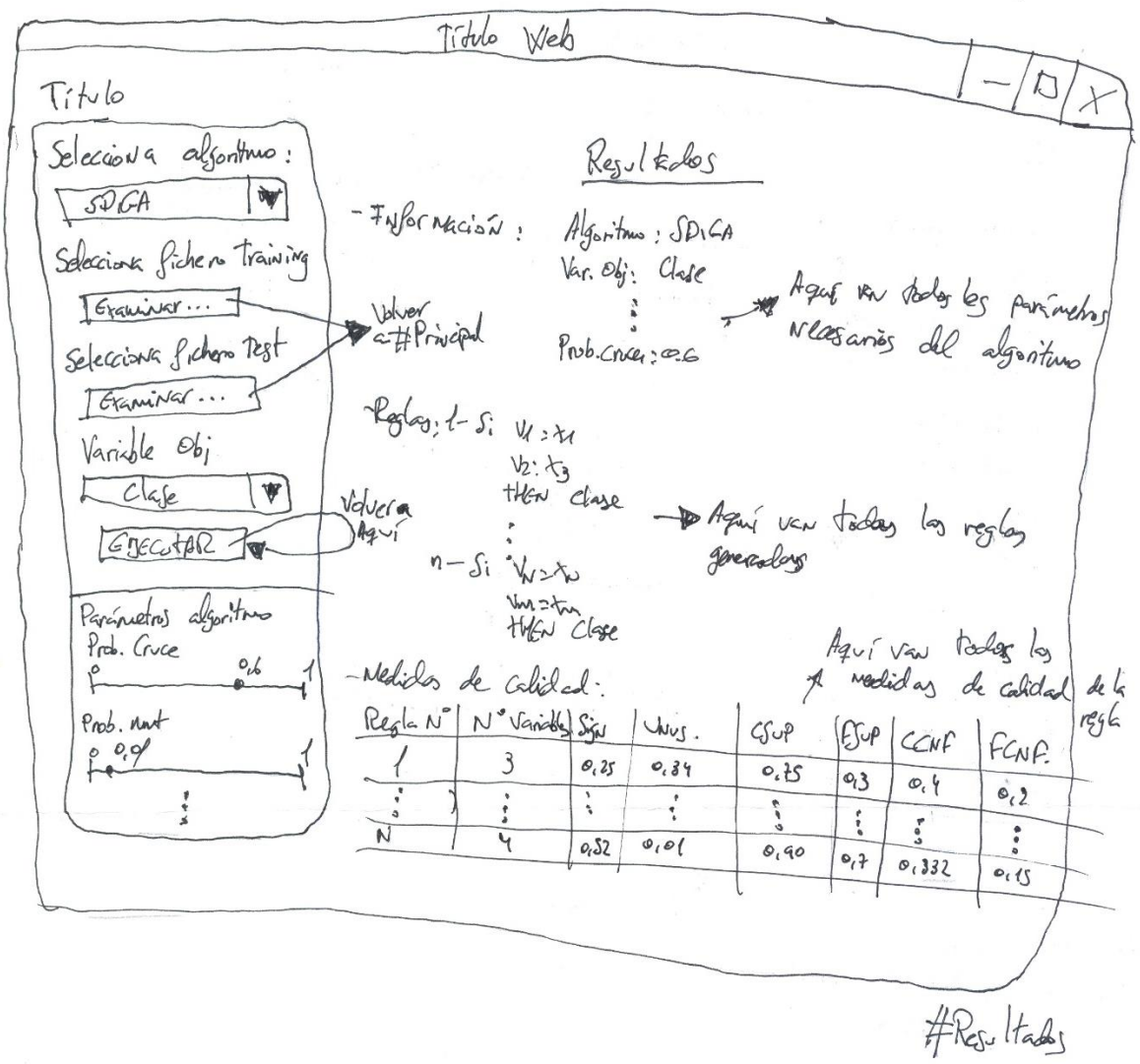


Figura 5.5. Storyboard, pantalla #Resultados

5.3. Implementación y pruebas

Para la implementación de la interfaz nos apoyaremos en un *framework* para la creación de aplicaciones web llamado *Shiny.*

Shiny es un paquete de R, creado y mantenido por RStudio, el cual nos permite crear aplicaciones web que utilicen todas las características que posee R, por lo que su potencia es muy alta. Además, *Shiny* permite crear sus aplicaciones directamente en R a través de un conjunto de funciones que permiten crear tanto la interfaz como la lógica del servidor, por lo que no es necesario el uso de HTML, CSS o JavaScript para crear una aplicación. Aunque no sea necesario, *Shiny* también nos permite en el caso de que lo deseemos añadir estilos con plantillas CSS, elementos directamente en HTML o funciones más complejas a través de JavaScript. Toda esta funcionalidad hace que sea un *framework* realmente potente a la vez que sencillo y completo si queremos personalizar nuestro sitio.

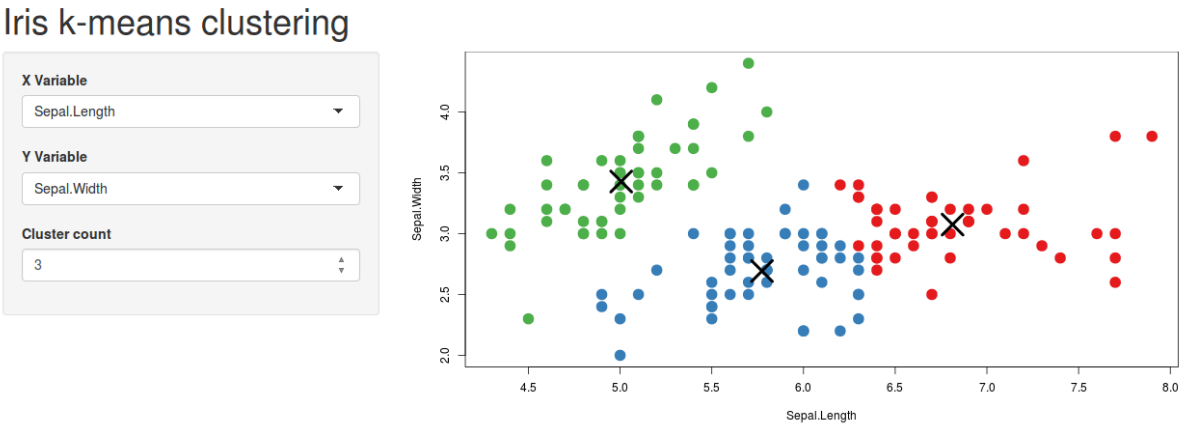


Figura 5.6. Aplicación Shiny simple

En el ejemplo de la Figura 5.6, se muestra un ejemplo de visualización del dataset iris usando k-means. En este ejemplo, no se ha utilizado nada de HTML, CSS o JavaScript, simplemente las funciones de R para visualizar la gráfica y las funciones para mostrar los selectores.

Todo esto, al ser un proyecto de RStudio, está completamente integrado en el IDE, por lo que crear, ejecutar, depurar, visualizar e incluso publicar en Internet nuestra aplicación es sencillo.

La verificación de los algoritmos ya ha sido realizada en la sección 4.3 por lo que no es necesario comprobarlo de nuevo. En este caso también se ha seguido un enfoque *TDD* por lo que los casos de prueba se han realizado antes incluso de la implementación, haciendo que la implementación tenga menos fallos y posibilitando el uso de herramientas automáticas de validación.

Capítulo 6:

Experimentación y conclusiones finales

6. Experimentación y conclusiones finales

En este capítulo final de la memoria se exponen una serie de experimentaciones con bases de datos públicas, con el fin de analizar los resultados obtenidos y se concluye con el trabajo futuro para la mejora de este proyecto.

6.1. Experimentación con bases de datos públicas

A continuación se detallarán una serie de expermientaciones llevadas a cabo con diferentes bases de datos públicas.

Características de la experimentación

6.1.1. Una vez hemos verificado el correcto funcionamiento de los algoritmos los ejecutaremos sobre distintas bases de datos públicas a modo de experimentación para analizar los resultados obtenidos.

Las bases de datos públicas las hemos obtenido del conjunto de datasets que nos ofrece KEEL [35]. Las características de dichas bases de datos se recogen en la siguiente tabla:

Nombre	Nº ejemplos	Nº Variables (E / R / N)	Nº valores del atributo de clase
Haberman	306	0/3/0	2
Iris	150	0/4/0	3
Monk-2	432	0/6/0	2
Tic-Tac-Toe	958	0/0/9	2
flare	1066	0/0/11	6
car	1728	0/0/6	4
australian	690	3/5/6	2
german	1000	0/7/13	6

Tabla 6.1. Características de las bases de datos para la experimentación

La experimentación se llevará a cabo siguiendo un esquema *k-fold Cross Validation.* Este esquema permite evaluar los resultados para garantizar que son independientes de la partición de entrenamiento-prueba que escojamos, por lo que se realizarán *k* ejecuciones del algoritmo y se mostrarán los resultados medios. El método divide el conjunto original de ejemplos en *k* subconjuntos de igual tamaño y en cada una de las diferentes ejecuciones, tomaremos un subconjunto diferente para

que sea el conjunto de prueba y el resto para el conjunto de entrenamiento. La Figura 6.1 que se muestra a continuación ejemplifica cómo se realizaría este método.

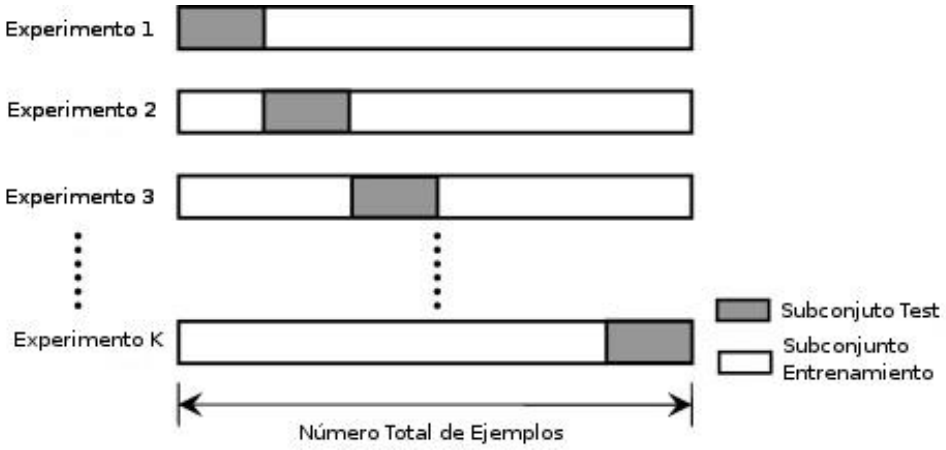


Figura 6.1. Esquema de validación cruzada

Para la experimentación utilizaremos un valor de *k* igual a cinco, por lo que nuestra experimentación constará de $5 \cdot 8 = 40$ ejecuciones por algoritmo y representación de las reglas, haciendo un total de 200 ejecuciones.

Los parámetros utilizados por los algoritmos durante la experimentación se recogen en las siguientes tablas:

a) SDIGA:

Parámetro	Valor
seed	1000
nLabels	3
nEval	10000
popLength	100
crossProb	0.6
mutProb	0.01
minConf	0.6
RulesRep	can (también dnf)
Obj1	CSUP(FSUP en *australian*)
Obj2	CCNF
Obj3	Null
w1	0.7
w2	0.3
w3	0.0
lSearch	No
targetClass	Null

Tabla 6.2. Parámetros utilizados para SDIGA

b) MESDIF:

Parámetro	Valor
seed	0
nLabels	3
nEval	10000
popLength	100
eliteLength	3
crossProb	0.6
mutProb	0.01
echo	no
RulesRep	can (también dnf)
Obj1	CSUP
Obj2	CCNF
Obj3	null
Obj4	null
targetClass	null

Tabla 6.3. Parámetros utilizados para MESDIF

c) NMEEF-SD:

Parámetro	Valor
Seed	1
nLabels	3
nEval	10000
popLength	51
crossProb	0.6
mutProb	0.1
minCnf	0.6
RulesRep	can
Obj1	CSUP
Obj2	CCNF
Obj3	Null
StrictDominance	yes
Diversity	crowding
ReInitCob	yes
porcCob	0.5
targetClass	null

Tabla 6.4. Parámetros utilizados para NMEEF-SD

Resultados obtenidos y análisis

En este apartado se recogen en tablas los resultados obtenidos, realizando un análisis comparativo de los mismos a continuación.

6.1.2.

6.1.2.1. Resultados usando representación canónica

Dataset	Algoritmo	NºReg.	NºVars	Cov.	Sign	WRacc	Acc.	CSupp	FSupp	Fconf	CConf
Haberman	SDIGA	**4**	**2**	0,3124	**7,7505**	**0,1466**	**0,829**	**0,9722**	**0,2684**	**0,80606**	**0,8549**
	MESDIF	6	2,766	**0,3621**	0,2573	0,0153	0,5464	0,9085	0,1981	0,5467	0,4920
	NMEEFSD	10,2	3,056	0,1902	0,3398	0,0356	0,6730	0,7026	0,1541	0,6163	0,6388
Iris	SDIGA	**3,2**	**2,340**	**0,3728**	**6,6172**	**0,1633**	0,7266	0,8799	**0,2539**	0,7615	0,8568
	MESDIF	9	2,800	0,3437	5,0973	0,1248	0,6308	**0,9600**	0,2163	0,6867	0,7186
	NMEEFSD	7,8	2,830	0,2744	6,4086	0,1576	**0,7324**	0,8867	0,2277	**0,8058**	**0,9039**
Monk-2	SDIGA	**4**	**2,000**	**0,3125**	**7,7506**	0,1467	0,8290	**0,9722**	**0,2685**	0,8060	0,8549
	MESDIF	6	2,366	0,2763	7,0371	0,1421	**0,8445**	0,9120	0,2055	**0,8347**	**0,8916**
	NMEEFSD	5,4	2,253	0,31166	7,03860	**0,15132**	0,82634	0,92828	0,22317	0,82255	0,85869
australian	SDIGA	**2**	**2,000**	**0,7580**	7,7505	0,0986	0,6690	0,9667	**0,4822**	0,6755	0,6733
	MESDIF	6	3,266	0,5024	6,8855	0,0821	0,7142	**0,9928**	0,3152	0,7632	0,7397
	NMEEFSD	52,2	4,937	0,2687	**11,9206**	**0,1396**	**0,8786**	0,8536	0,2164	**0,9091**	**0,9129**
german	SDIGA	**3**	3,500	**0,4933**	0,5656	0,0184	0,5972	0,8000	**0,2502**	0,6225	0,6350
	MESDIF	6	**3,000**	0,3933	1,1866	0,0411	0,6252	**0,9840**	0,2180	0,6475	0,6682
	NMEEFSD	37	4,405	0,2856	**3,1961**	**0,0905**	**0,7389**	0,8030	0,2096	**0,7450**	**0,7624**
car	SDIGA	**6**	**2,000**	**0,3182**	**17,7429**	**0,1162**	**0,5184**	**0,8206**	**0,1744**	**0,5275**	**0,5275**
	MESDIF	12	3,599	0,14704	8,47893	0,05418	0,42807	0,68343	0,07682	0,41671	0,41671
	NMEEFSD	14	4,950	0,0324	4,6652	0,0204	0,3714	0,3744	0,0279	0,3304	0,3304
flare	SDIGA	**6**	**4,032**	0,2916	1,5951	0,0067	0,1675	0,3124	0,0521	0,0597	0,0597
	MESDIF	18	4,188	**0,32464**	**19,6159**	**0,05915**	0,41211	**0,99813**	**0,11148**	0,48434	0,48434
	NMEEFSD	101,2	5,491	0,06959	17,8282	0,04877	**0,48451**	0,66047	0,05808	**0,65190**	**0,65190**
Tic-tac-toe	SDIGA	**4**	**2,000**	**0,4301**	2,8897	**0,1703**	0,6949	**0,8531**	0,3045	0,6995	0,6995
	MESDIF	6	3,066	0,19710	**4,32935**	0,0977	**0,75481**	0,60129	**0,14543**	**0,79527**	**0,79527**
	NMEEFSD	10,4	3,163	0,1529	3,3797	0,0873	0,7448	0,5251	0,1158	0,7731	0,7731

Tabla 6.5. Resultados de las experimentaciones utilizando representación canónica

Observando detenidamente los resultados obtenidos, llegamos a las siguientes conclusiones:

- De media, la cantidad de reglas generadas por SDIGA es menor que la de NMEEF-SD. Con MESDIF este valor no se puede comparar puesto que el número de reglas que obtenemos viene prefijado cuando seleccionamos el valor de la población élite.
- Sin embargo, en interpretabilidad de las reglas obtenidas SDIGA sí es mejor en la mayoría de reglas.
- Al tener menos variables en el antecedente, las reglas obtenidas son más generales y, por lo tanto, su antecedente cubrirá bastantes ejemplos. Por esta razón el valor de cobertura (coverage) es normalmente más alto en SDIGA que en el resto.
- En cuanto a la significancia de las reglas generadas, no existe un algoritmo dominante en este caso y las diferencias entre ellos no son lo suficientemente grandes como sacar una conclusión clara.
- Para el valor de atipicidad, hay igualdad entre NMEEF-SD y SDIGA en cuanto a la atipicidad de las reglas generadas.
- En cuanto a soporte: a pesar de que SDIGA obtiene el mismo número de mejores resultados que MESDIF, los mejores resultados de MESDIF son mucho mejores que los de SDIGA y NMEEF-SD. Seguramente la razón de este hecho se deba al uso del operador de truncado de MESDIF que busca diversificar el número de reglas cubriendo el máximo espacio del frente de Pareto.
- En cuanto hablamos de la confianza: NMEEF-SD es el que obtiene mejores resultados y no es de extrañar ya que por lo que arroja la tabla, las reglas generadas por este algoritmo son muy específicas ya que tenemos muchas reglas con muchas variables de media.

6.1.2.2. Resultados para representación DNF

Dataset	Algoritmo	NºReg.	NºVars	Cov.	Sign	WRacc	Acc.	CSupp	FSupp	Fconf	CConf
Haberman	SDIGA	**3,0000**	**2,126**	**0,8054**	0,0543	**0,0113**	**0,5578**	0,9739	**0,3891**	0,5295	**0,5353**
	MESDIF	6,0000	3,166	0,3589	**0,2128**	0,0106	0,5420	**0,9903**	0,1554	**0,5538**	0,4929
Iris	SDIGA	**3,0000**	**2,360**	**0,4600**	**7,9696**	**0,1778**	**0,6923**	0,9933	**0,2545**	**0,6815**	**0,7879**
	MESDIF	7,4000	3,625	0,3290	5,4260	0,1234	0,6177	**1,0000**	0,1938	0,6272	0,7218
Monk-2	SDIGA	**2,0000**	**2,000**	**0,8333**	2,6710	0,0832	0,6078	0,9722	**0,4841**	0,6125	0,6118
	MESDIF	5,8000	3,066	0,5894	**10,9277**	**0,1420**	**0,7827**	**1,0000**	0,3840	**0,7475**	**0,7992**
australian	SDIGA	**2,0000**	**2,100**	**0,7601**	**7,7485**	**0,0983**	0,6678	0,9667	**0,5379**	0,7321	0,6721
	MESDIF	6,0000	5,066	0,4618	7,1912	0,0796	**0,7127**	**1,0000**	0,2679	**0,7695**	**0,7024**
german	SDIGA	**2,4000**	**2,666**	**0,8630**	0,0013	0,0011	0,5001	**0,9970**	**0,4324**	0,4668	0,4334
	MESDIF	6,0000	3,666	0,3928	**1,1861**	**0,0359**	**0,6014**	0,9780	0,2149	**0,6028**	**0,5904**
car	SDIGA	**5,0000**	**3,600**	**0,3889**	**26,8293**	**0,1076**	**0,5967**	**1,0000**	**0,2867**	0,6189	0,6189
	MESDIF	12,0000	4,685	0,1939	14,7939	0,0625	0,5839	0,9155	0,1384	**0,7053**	**0,7053**
flare	SDIGA	**8,0000**	**4,833**	**0,3325**	**46,0503**	**0,0852**	**0,4476**	0,9831	**0,1367**	**0,4680**	**0,4680**
	MESDIF	17,4000	6,103	0,2340	28,5583	0,0648	0,4180	**0,9934**	0,1035	0,4375	0,4375
Tic-tac-toe	SDIGA	**3,0000**	**2,000**	**0,7704**	1,3018	0,1053	0,5897	**0,9499**	**0,4440**	0,5912	0,5914
	MESDIF	6,0000	3,266	0,4020	**6,3074**	**0,1110**	**0,7124**	0,9040	0,2486	**0,7301**	**0,7301**

Tabla 6.6. Resultados de las experimentaciones utilizando representación DNF

Observando los resultados obtenidos las conclusiones a las que se han llegado son las siguientes:

- La interpretabilidad de las reglas obtenidas es, en todos los casos de la experimentación, mejores en SDIGA, al igual que su cobertura, reforzando la idea de la generalidad de las reglas.
- La significancia obtenida por MESDIF es muy superior en muchas ocasiones que la obtenida por SDIGA y en el caso de que SDIGA supere a MESDIF, este obtiene un resultado solo ligeramente superior, por lo que las reglas obtenidas por MESDIF son mucho más novedosas que las generadas por SDIGA. Este resultado es posible que determine el uso de MESDIF antes que SDIGA, ya que lo que buscamos en descubrimiento de subgrupos son reglas novedosas, atípicas. Este resultado es menos notable (la diferencia entre ellos es ligeremente inferior) en la medida de atipicidad, pero la conclusión que obtenemos es la misma que para el valor de significancia.

- Los valores de soporte son muy similares en soporte nítido, este hecho puede deberse a la falta de suficientes ejemplos en estos datasets. Sin embargo, en el soporte difuso, el valor es mucho mejor en SDIGA que para MESDIF, esto significa que las reglas generadas por SDIGA son mas identificativas con los ejemplos por que tienen mayor valor de *APC*.
- También de manera lógica, si una regla tiene mucho soporte es muy probable que su confianza decaiga. Esto ocurre en SDIGA, cuyos valores de confianza normalmente son más bajos que los de MESDIF al tratarse de reglas más generales y, por lo tanto, más imprecisas.

6.2. Conclusiones finales y trabajo futuro

Una vez concluido todo el desarrolo de nuestro proyecto expondremos a continuación una serie de conclusiones relacionadas con los objetivos establecidos al principio del mismo y su grado de cumplimiento, así como posible trabajo futuro relacionado con este proyecto.

6.2.1. *Conclusiones*

Tras todo el desarrollo de nuestro proyecto, las conclusiones que obtenemos son las siguientes:

- Se ha realizado un estudio del estado del arte en minería de datos, profundizando en la tarea de descubrimiento de subgrupos.
- Se han realizado diferentes tareas de ingeniería del software para realizar el análisis y diseño de una librería de algoritmos para el lenguaje de programación R, en el cual se han implementado tres algoritmos de descubrimiento de subgrupos.
- También se han realizado las diferentes tareas de ingeniería del software relacionadas con el análisis y diseño de una plataforma web simple para el uso de los algoritmos implementados siguiendo para ello un enfoque de diseño orientado al usuario.
- Se han realizado diferentes experimentaciones con los algoritmos implementados en bases de datos públicas, obteniendo resultados útiles y aplicables. También se ha realizado un análisis de las medidas de calidad

obtenidas por cada algoritmo para intentar buscar el algoritmo más adecuado para un objetivo dado.

Para el cumplimiento de estos objetivos iniciales, se han desarrollado tres algoritmos de descubrimientos de subgrupos y una plataforma web que permiten obtener subgrupos de una manera cómoda y simple. Además, tanto los algoritmos como la plataforma web se han unificado en un único paquete de R para su fácil utilización por toda persona que utilice R.

Para concluir cabe destacar que se han cumplido todos los objetivos definidos en este proyecto. Como conclusión personal me gustaría añadir que este proyecto me ha aportado gran cantidad de nuevo conocimiento, como aprender un nuevo lenguaje de programación con un paradigma totalmente diferente al visto en el Grado y que ha sido un auténtico reto enfrentarse a él. También este proyecto me ha servido para el afianciamiento de conocimientos sobre todo de la propia ingeniería del software, así como en minería de datos, el cual me ha permitido profundidar en un tarea bastante nueva y en la que hay mucha investigación activa en la actualidad.

6.2.2. *Trabajo Futuro*

Las líneas de trabajo futuro que deja este proyecto abiertas son:

- Inclusión de nuevos algoritmos evolutivos de descubrimiento de subgrupos de la bibliografía especializada, como por ejemplo FuGePSD (Fuzzy Genetic Programming-based for Subgroup Discovery) [36].
- Mayor optimización de los algoritmos existente. Esta puede realizarse de dos formas, o bien reorientando el algoritmo para aprovechar al máximo la vectorización o implementando el algoritmo en C++ utilizando el paquete y la librería *Rcpp.*
- Facilitar una personalización mayor de los algoritmos permitiendo definir diferente cantidad de etiquetas difusas según la variable que estemos utilizando así como la posibilidad de manejar mayor cantidad de formatos de datasets,

como por ejemplo el .arff de WEKA [8] o la posibilidad de convertir estos formatos a formato KEEL.

- Añadir mayor funcionalidad a la plataforma web, principalmente en temas de análisis exploratorio ya que R es una potente herramienta de análisis exploratorio, incluyendo por ejemplo, diagramas de dispersión, mayor cantidad de medidas estadísticas, etc.
- Preparación del paquete para alojarlo en CRAN (The Comprenhensive R Archive Network), el lugar donde se almacenan los paquetes que instalamos desde R. Actualmente el paquete solo está disponible de manera local o desde GitHub (dependiendo de un paquete para instalarlo desde esta plataforma) y para alojarlo en CRAN debe de seguir de forma estricta las características que se indican en [37] ya que son muy exigentes en cuanto a su cumplimiento.

Bibliografía

7. Bibliografía

[1] «The Internet In Real Time,» 24 04 2015. [En línea]. Available: http://pennystocks.la/internet-in-real-time/.

[2] C. J. Carmona, S. Ramirez-Gallego, F. Torres, E. Bernal, M. J. del Jesus y S. García , «Web usage mining to improve the design of an e-commerce website: OrOliveSur. com,» *Expert Systems with Applications,* pp. 11243-11249, 2012.

[3] K.-S. Leung, K. H. Lee, J.-F. Wang, E. Y. Ng, H. L. Y. Chan, S. K. Tsui, T. S. Mok, P. C.-H. Tse y J.-Y. Sung, «Data Mining on DNA Sequences of Hepatitis B Virus,» *IEEE/ACM Transactions on computational biology and bionformatics,* 2011.

[4] T. Mullen y N. Collier, «Sentiment analysis using support vector machines with diverse information,» 2004.

[5] U. Fayyad, G. PIstetsky-Shapiro y P. Smyth, «From Data Mining to Knowledge Discovery in Databases,» 1996.

[6] S. Wrobel, «Inductive logic programming for knowledge discovery in databases,» de *Relational Data Mining*, Springer, 2001, pp. 74-101.

[7] EuropaPress, «Expertos en análisis de datos aseguran que la demanda de científicos en este área "será ingente en el futuro",» 25 08 2014. [En línea]. Available: http://www.europapress.es/andalucia/noticia-expertos-analisis-datos-aseguran-demanda-cientificos-area-sera-ingente-futuro-20140825184716.html.

[8] M. Hall, F. Eibe, G. Holmes, B. Pfahringer, P. Reutemann y I. H. Witten, «The WEKA Data Mining Tool: An update,» *SIGKDD Explorations,* vol. 11, nº 1, 2009.

[9] L. S. S. G. M. d. J. S. V. J. G. J. O. C. R. J. B. V. R. J. F. F. H. J. Alcalá-Fdez, «KEEL: A Software Tool to Assess Evolutionary Algorithms to Data Mining Problems,» de *Soft Computing*, 2009, pp. 307-318.

[10] M. Atzmueller y F. Lemmerich, «VIKAMINE - Open-Source Subgroup Discovery, Pattern Mining and Analytics,» de *Machine Learning and Knowledge Discover in Databases*, 2012, pp. 842-845.

[11] R Core Team, R: A Language and Environment for Statistical Computing, Vienna, Austria, 2015.

[12] «Languages for analytics/data mining (Aug 2014),» 19 06 2015. [En línea]. Available: http://www.kdnuggets.com/polls/2015/analytics-data-mining-data-science-software-used.html.

[13] M. del Jesus, P. González, F. Herrera y M. Mesonero, «Evolutionary fuzzy rule induction process for subgroup discovery: a case study in marketing,» *IEEE Trans Fuzzy Syst.,* pp. 15(4):578-592, 2007.

[14] F. Berlanga, M. J. Del Jesus, P. González, F. Herrera y M. Mesonero, «Multiobjective Evolutionary Induction of Subgroup Discovery Fuzzy Rules: A Case Study in Marketing,» 2006.

[15] C. J. Carmona, P. González, M. J. del Jesús y F. Herrera, NMEEF-SD: Non-dominated Multi-objective Evolutionary algorithm for Extracting Fuzzy rules in Subgroup Discovery, 2010.

[16] C. J. Carmona, P. González, M. J. del Jesus, N.-A. M. y L. Jiménez-Treviño, «Evolutionary fuzzy rule extraction for subgroup discovery,» *Soft Computing,* vol. 15, pp. 2435-2448, 2011.

[17] J. Quinlan, C4.5: programs for machine learning, 2014.

[18] J. Hartigan y M. Wong, «Algorithm AS 136: A k-means clustering algorithm,» *Applied statistics,* 1979.

[19] R. Agrawal, T. Imieliski y A. Swami, «Mining association rules between sets of items in large databases,» *Proceedings of the 1993 ACM SIGMOD international conference on management of data,* pp. 207-216, 1993.

[20] F. Herrera, M. J. del Jesus, P. González y C. J. Carmona, «An overview on subgroup discovery: foundations and applications,» 2010.

[21] W. Kloesgen, «Explora: a multipattern and multistrategy discovery assistant.,» *Advances in Knowledge discovery and data mining. America Association for Artificial Intelligence,* pp. 249-271, 1996.

[22] S. Wrobel, «An algorithm for multi-relational discovery of subgroups,» *Proceedings of the 1st European symposium on principles of data mining and knowledge discovery,* vol. 5651, pp. 265-274, 1997.

[23] D. Gamberger y N. Lavrac, «Expert-guided subgroup discovery: methodology and application,» *J Artif Intell Res,* pp. 501-527, 2002.

[24] N. Lavrac, B. Kavsek, P. Flach y L. Todorovski, «Subgroup discovery with CN"-SD,» *J Mach Learn Res,* pp. 5:153-188, 2004.

[25] P. Clark y T. Niblett, «The CN2 induction algorithm,» *MAchine Learning,* vol. 3, nº 4, pp. 261-283, 1989.

[26] B. Kavsek y N. Lavrac, «APRIORI-SD: adapting association rule learning to subgroup discovery,» *Appl Artif Intell,* pp. 20:543-583, 2006.

[27] V. Jovanoski y N. Lavrac, «Classification Rule Learning with APRIORI-C,» de *Progress in Artificial Intelligence*, Springer, 2002, pp. 44-51.

[28] M. Atzmueller y F. Puppe, «SD-Map-a fast algorithm for exhaustive subgroup discovery,» *Proceeding of th 17th European conference on machine learning and 10th European conferen on principles and practice of knowledge discovery in databases,* vol. 4213. Springer LNCS , pp. 6-17, 2006.

[29] J. Han, H. Pei y Y. Yin, «Mining Frequent Patterns without Candidate Generation,» de *Proc. Conf. on the Management of Data*, Dallas, TX, 2000.

[30] M. del Jesus, P. González y F. Herrera, «Multiobjective genetic algorithm for extracting subgroup discovery fuzzy rules.,» *Proceedings of the IEEE symposium on computational intelligence in multi-criteria decision making,* pp. 50-57, 2007.

[31] E. Zitzler, M. Laumanns y L. Thiele, SPEA2: Improving the Strength Pareto Evolutionary Algorithm, 2001.

[32] K. Deb, S. Agrawal, A. Pratap y T. Mayerivan, A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II, 2000.

[33] P. González, Aprendizaje Evolutivo de Reglas Difusas para Descripción de Subgrupos, 2007.

[34] C. Carmona, P. González , M. del Jesus y F. Herrera, «An analysis of evolutionary algorithms with different types of fuzzy rules in subgroup discovery,» de *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, ICC Jeju, Jeju Island, Korea, 2009.

[35] A. F. J. L. J. D. S. G. L. S. F. H. J. Alcalá-Fdez, «KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework,» *Journal of Multiple-Valued Logic and Soft Computing ,* pp. 255-287, 2011.

[36] C. Carmona, V. Ruíz-Rodado, A. Weber, M. Grootyeld, P. González y D. Elizondo, «A fuzzy genetic programming-based algorithm for subgroup discovery and the application to one problem of pathogenesis of acute sore throat conditions in humans,» *Information Sciences,* vol. 298, pp. 180-197, 20 3 2015.

[37] The R Core Team, Writing R Extensions, 2014.

[38] M. Hellmann, «Fuzzy Logic Introduction».

[39] F. Charte, Análisis exploratorio y visualización de datos con R, 2014.

[40] L. Scrucca, «GA: A Package for Genetic Algorithms in R,» *Journal of Statistical Software,* 2013.

[41] E. Paradis, R para Principiantes, 2002.

[42] P. Burns, The R Inferno, 2011.

[43] H. Wickham, Advanced R, 2014.

[44] G. Grolemund, Hands-On Programming with R, O´Reilly, 2014.

[45] H. Wickham, R Packages, O'Reilly, 2015.

Anexo I. Manual de Instalación

Anexo I Manual de instalación

A continuación mostramos cómo instalar las herramientas necesarias para ejecutar este proyecto, tanto en un sistema operativo Windows como en un sistema operativo Linux, en particular, Ubuntu.

Instalación de R

En este apartado se explica cómo instalar R en nuestro sistema, ya sea desde Windows o desde Ubuntu.

AI.1.1. Windows

El manual que exponemos se llevará a cabo en Windows 8.1, ya que es el sistema que actualmente estamos utilizando. Sin embargo, la instalación es similar para versiones anteriores, variando únicamente el estilo de las ventanas de las imágenes de ejemplo.

AI.1.2. Requisitos previos

No es necesario ningún requisito previo para instalar R en nuestro sistema Windows, aunque es recomendable tener una conexión a Internet para poder descargarse la última versión de la herramienta.

AI.1.3. Instalación

Si tenemos una conexión a Internet, es posible instalar la última versión desde este enlace: http://cran.r-project.org/bin/windows/base/, para descargar la última versión, simplemente tenemos que pinchar en el enlace "Download R 3.2.1 for Windows"[4].

En el caso de no disponer de conexión a Internet, en el CD-ROM en el que se entrega esta memoria disponemos de una carpeta llamada "**Instalacion**". Esta carpeta tiene un fichero que se llama "R-3.2.1-win.exe" el cual posee la versión 3.2.1 de R.

[4] Actualmente, la versión más reciente de R es la 3.2.1 y es posible que varíe a lo largo del tiempo.

Una vez tenga el fichero de instalación, lo abriremos. Al abrir, se nos mostrará una ventana con una advertencia de seguridad. Haremos clic en “Ejecutar”.

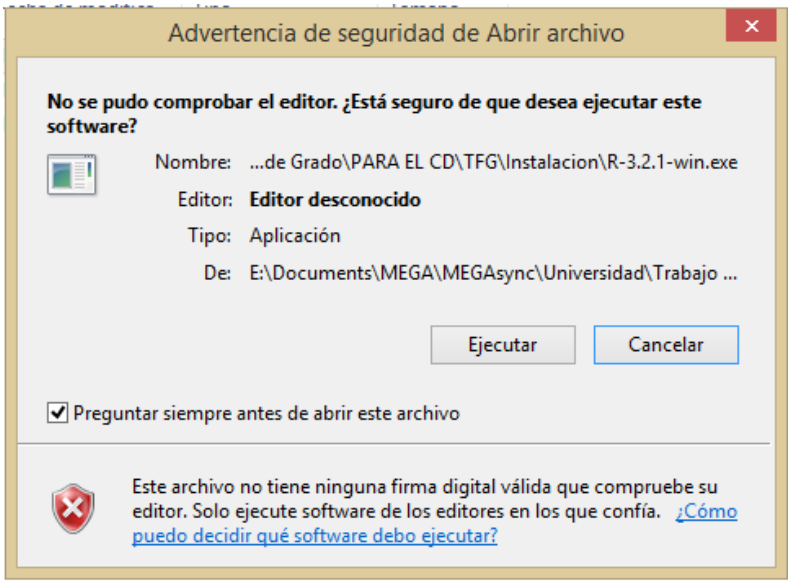


Figura A I.1. Advertencia de seguridad al abrir archivo

A continuación, nos pedirá que seleccionemos idioma. En nuestro caso, seleccionamos Español y pulsaremos **Aceptar**. Después pulsaremos en “**Siguiente**”. La siguiente pantalla nos muestra el acuerdo de licencia de uso de R. Una vez lo leamos y estemos de acuerdo con él, pulsaremos “**Siguiente**”. La próxima pantalla nos muestra dónde queremos instalar R en nuestro ordenador. Si quisiéramos elegir una carpeta diferente a la que se encuentra por defecto, pulsaremos en “**Examinar…**”.

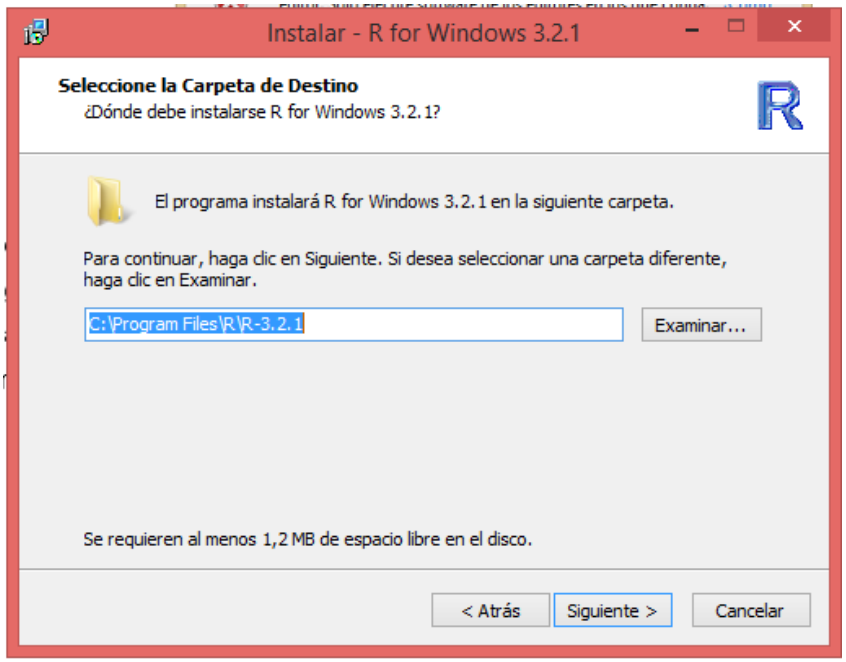


Figura A I.2. Selección de carpeta de destino de R

Una vez elegida nuestra carpeta de instalación, pulsaremos en "**Siguiente**". A continuación tenemos que seleccionar qué componentes queremos instalar. Dejaremos la selección por defecto, por lo que pulsaremos en **"Siguiente".**

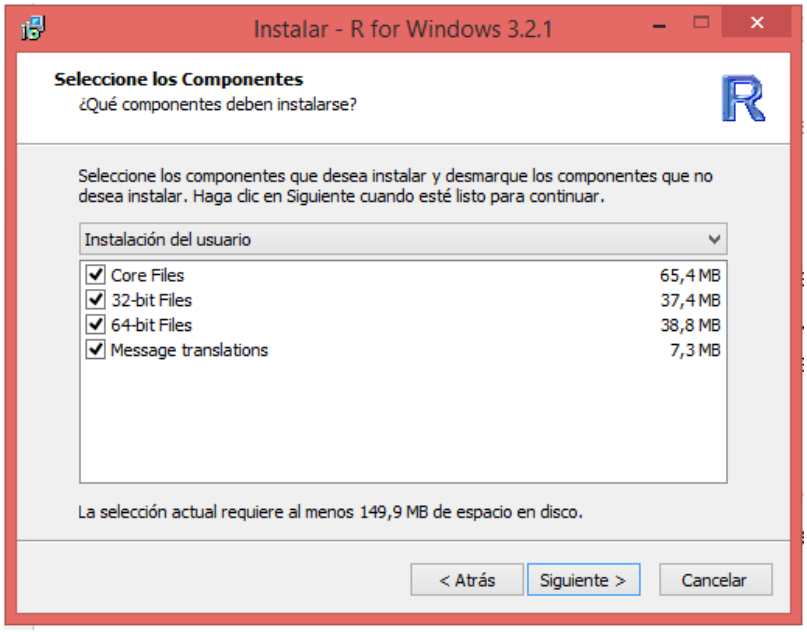


Figura A I.3. Instalación de componentes adicionales

Después de esto, la siguiente pantalla nos indica si queremos utilizar opciones de configuración de R. Para el uso de nuestra librería no es necesario opciones de configuración adicionales, por lo que marcamos “no” y pulsamos en “**Siguiente**”. La

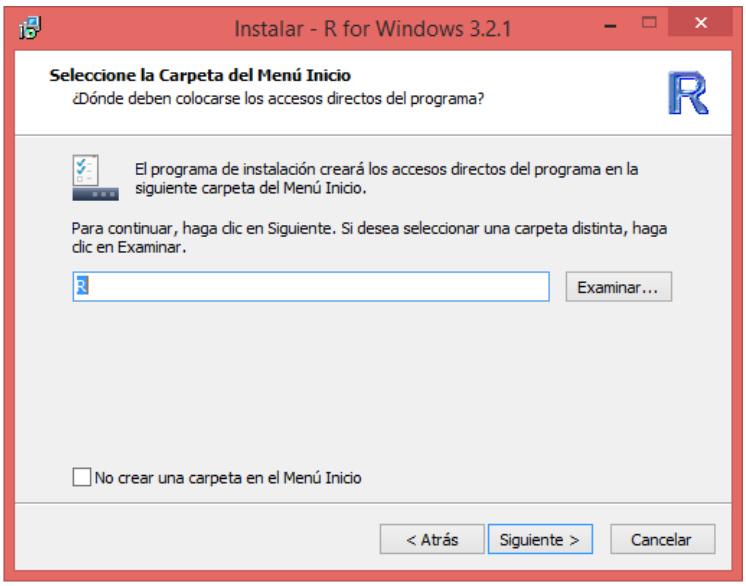


Figura A I.5. Selección de carpeta del Menú de Inicio

próxima pantalla indica dónde se ubicaran los accesos directos del menú de inicio. Si no tenemos ninguna configuración especial de nuestro sistema Windows, se pueden dejar los valores por defecto. Una vez esté la configuración a nuestro gusto, podemos pulsar en “**Siguiente**”.

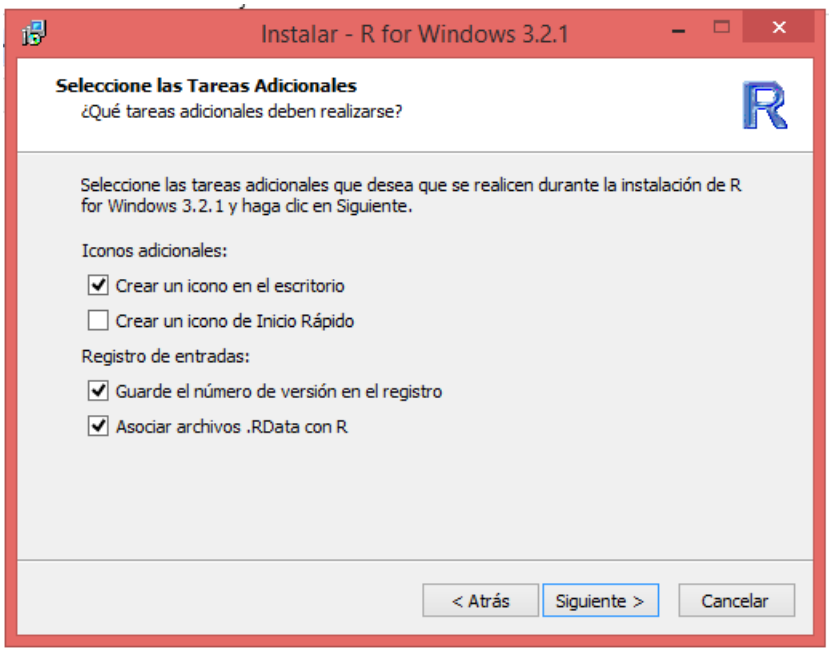


Figura A I.4. Selección de tareas adiciones en la instalación

A continuación se nos muestran cuatro casillas para marcar en el caso de que queramos tener iconos adiciones tales como accesos directos en el Escritorio. Una vez seleccionemos las casillas que creamos convenientes, pulsaremos "**Siguiente**" y el proceso de instalación comenzará. Una vez termine, pulsaremos **"Finalizar"** para terminar con la instalación de nuestro sistema.

AI.1.4. Ubuntu

A continuación se muestra cómo instalar R en un sistema Ubuntu, en particular, Ubuntu 14.04 LTS (Trusty Tahr). Esta se realiza mediante línea de comandos, por lo que será necesario ser un usuario con permisos root, ya que se usuará la orden `sudo`.

AI.1.5. Requisitos Previos

Para la instalación de R en Ubuntu será necesario requerir de una conexión a Internet. Además de una cuenta de administrador.

AI.1.6. Instalación

Para instalar R en ubuntu, deberemos de añadir una entrada en el archivo "/etc/apt/sources.list". Este proceso lo podemos hacer gráficamente desde la aplicación "Software y actualizaciones". Una vez abierta, pulsamos el botón "**Añadir**" de la pestaña "Otro software". Una vez hecho esto, nos aparecerá la siguiente ventana:

Una vez aquí, tanto si modificamos el fichero “sources.list” a mano, como si lo

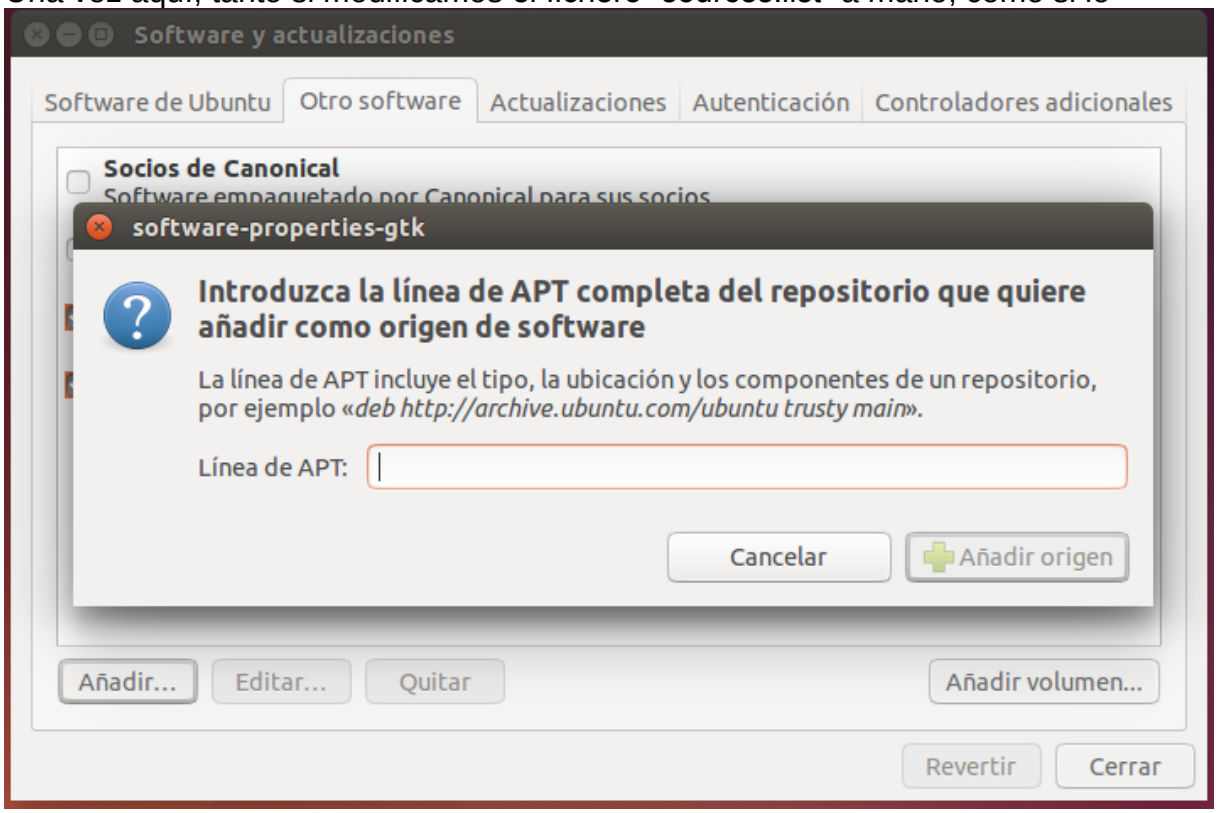


Figura A I.6. Adición de línea de APT para instalación de R

modifcamos en esta ventana, la línea de APT que debemos de añadir será la siguiente:

```
deb http://cran.rstudio.com/bin/linux/ubuntu trusty/
```

Una vez añadida, pulsamos en “Añadir origen” o guardamos el fichero “sources.list”.

Añadida la entrada de APT, nos dirigiremos a la *shell* y teclearemos los siguientes comandos:

```
sudo apt-get update
```

```
sudo apt-get install r-base
```

Y para finalizar:

```
sudo apt-get install r-base-dev
```

Una vez terminados los comandos tendremos R instalado en nuestro sistema y se podrá utilizar desde la línea de comandos. Para ello abriremos la *shell* y escribimos 'R' (sin comillas) y se nos abrirá una sesión en donde podremos introducir comandos de R.

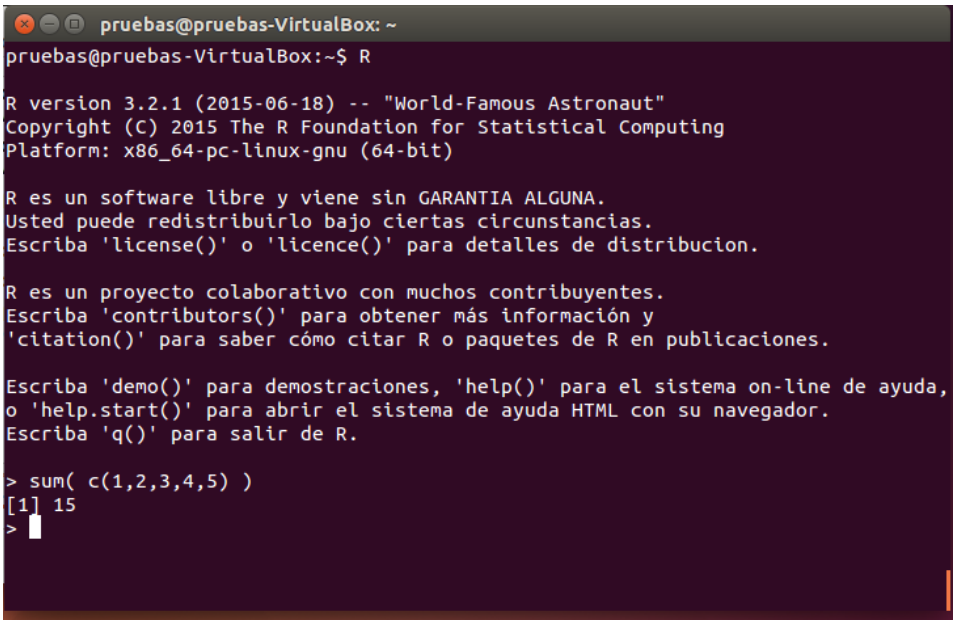


Figura A I.7. Consola de R en línea de comando de Ubuntu

Instalación de RStudio.

En este apartado se explica cómo instalar el entorno de desarollo RStudio, explicado con anterioridad en esta memoria. Para la utilización de la librería no es necesario tener que instalar este entorno de desarrollo, pero recomendamos encarecidamente que se instale para tener un mayor control de la sesión de R.

Para la descarga de la última versión de RStudio, podemos visitar la página http://www.rstudio.com/products/rstudio/download/ y descargamos el archivo correspondiente a nuestro sistema operativo.

AI.1.7. Requisitos Previos.

Para que RStudio funcione correctamente, es necesario tener instalado R en nuestro sistema con una versión igual o superior a la 2.11.1.

AI.1.8. Windows

Si no podemos obtener de Internet la última versión del entorno de desarrollo, tenemos, en el CD-ROM en el que se entrega esta memoria, dentro de la carpeta "**Instalacion**", un fichero llamado "**RStudio-Windows.exe**". Abrimos este fichero y pulsaremos en "**Siguiente**". A continuación se nos pregunta por la ruta de instalación. Modificaremos la ruta si lo deseamos y pulsaremos de nuevo en "**Siguiente**"

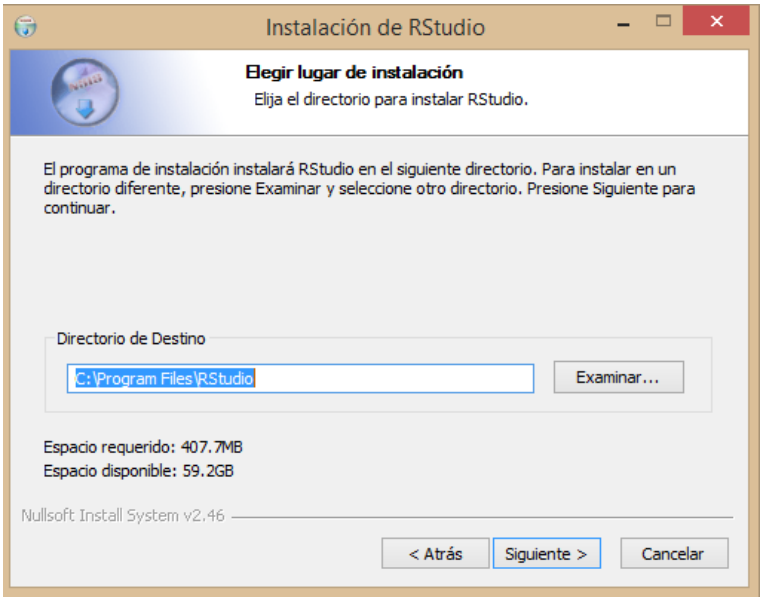


Figura A I.8. Selección de carpeta de destinio de RStudio

La posterior pantalla muestra el nombre que le queremos dar a la carpeta en el menu de inicio para crear accesos directos. Es recomendable dejarlo con el nombre por defecto, por lo que mantenemos el valor por defecto y pulsamos en "**Instalar**" y comenzará la instalación. Una vez terminado podremos ejecutar RStudio buscándolo en el menú de inicio de Windows.

AI.1.9. Ubuntu

Si no podemos obtener de Internet la última versión del entorno de desarrollo, tenemos, en el CD-ROM en el que se entrega esta memoria, dentro de la carpeta "**Instalacion**", un fichero llamado "**RStudio - Ubuntu.deb**". Al abrir este fichero se nos abrirá el *Centro de software de Ubuntu***,** verificamos que se va a instalar RStudio y pulsamos en instalar. Nos pedirá contraseña de root.

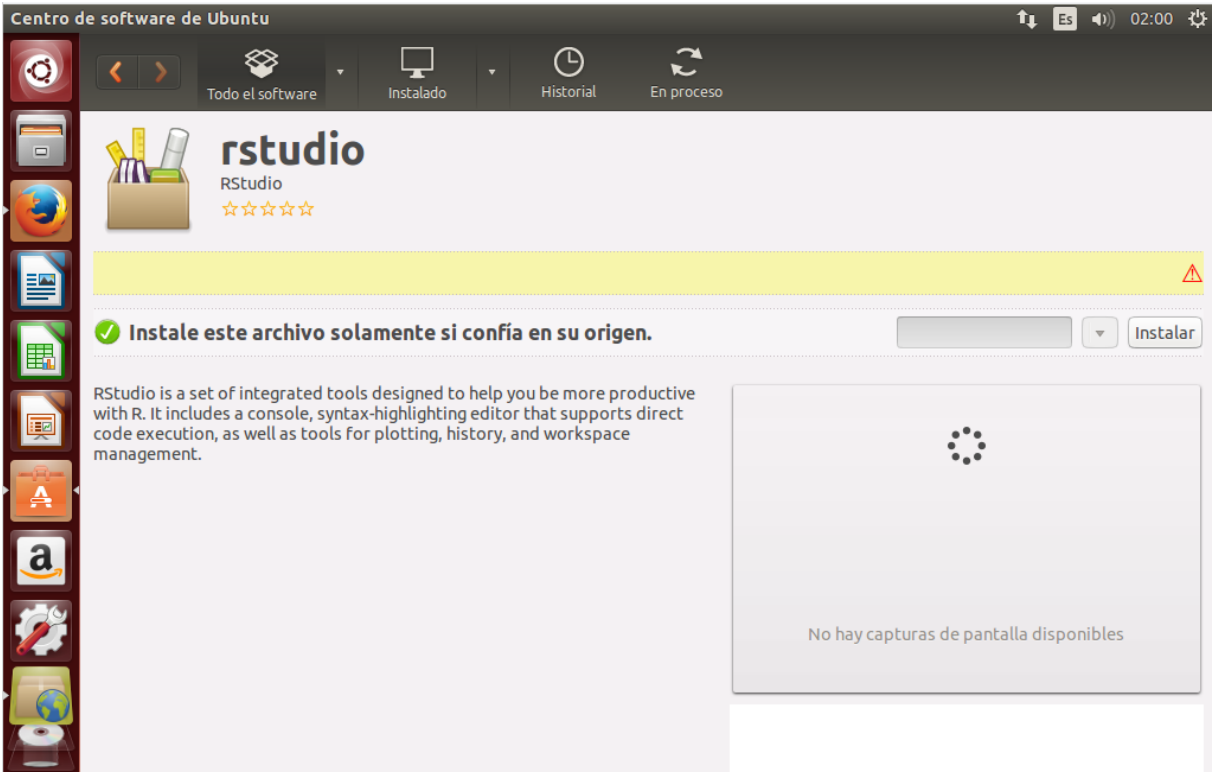


Figura A I.9. Instalación de RStudio desde el Centro de Software de Ubuntu

Instalación del paquete "SDR"

En el CD-ROM que acompaña a esta memoria se encuentra, aparte del código fuente, el paquete comprimido y listo para ser instalado en R. Este paquete se encuentra en el directorio "**Paquete R para Instalar**" y el fichero llamado "**SDR_1.0.tar.gz**" es nuestro paquete.

Adicionalmente, y debido a las mejoras que se llevarán a cabo en un futuro en este paquete, se dispone de un repositorio público en GitHub (https://github.com/aklxao2/SDR) para clonar el código y aportar a su desarrollo, descargarse la versión más reciente del paquete, el cual se encuentra dentro de la

carpeta "***package***", o instalar directamente el paquete en R que se encuentra en la rama master de GitHub utilizando la función del paquete "*devtools"*:

```
devtools::install_github('aklxao2/SDR')
```

Para instalar nuestro paquete, sino lo hemos realizado ya con `install_github()`, deberemos abrir R (en el ejemplo usaremos RStudio por comodidad, pero no es necesario). Una vez abierto, deberemos desplazarmos hasta el directorio donde tengamos nuestro fichero ".tar.gz" correspondiente a nuestro paquete utilizando la función `setwd()` y poniendo la ruta de la carpeta que contiene al fichero como parámetro. A continuación, para instalar nuestro paquete y suponiendo que se llama "**SDR_1.0.tar.gz**", utilizaremos la siguiente función para instalarlo en R:

```
install.packages("SDR_1.0.tar.gz", repos = NULL, type = "source")
```

```
> setwd("~/MEGA/MEGAsync/Universidad/Trabajo Fin de Grado/PARA EL CD/TFG/Paquete R para Instalar")
> install.packages("SDR_1.0.tar.gz", repos = NULL, type = ""source")
Installing package into 'E:/Documents/R/win-library/3.2'
(as 'lib' is unspecified)
* installing *source* package 'SDR' ...
** R
** data
*** moving datasets to lazyload DB
** preparing package for lazy loading
Creating a generic function for 'print' from package 'base' in package 'SDR'
Creating a generic function for 'summary' from package 'base' in package 'SDR'
Creating a generic function for 'plot' from package 'graphics' in package 'SDR'
** help
*** installing help indices
** building package indices
** testing if installed package can be loaded
*** arch - i386
*** arch - x64
* DONE (SDR)
>
```

Figura A I.10. Instalación del paquete de R en formato .tar.gz

Anexo II. Manual de Usuario

Anexo II Manual de Usuario

En este anexo presentamos un manual para el uso adecuado de nuestro sistema, tanto del paquete, como la interfaz web.

Manual de Uso del paquete.

Para poder utilizar las funciones existentes en nuestro paquete podemos realizarlo de dos maneras posibles:

- Enlazando el paquete en el llamado "*search path*" de R.
- Utilizando el espacio de nombres de nuestro paquete.

Para utilizar una función con el espacio de nombres, esta debe ir precedida de "SDR::" (sin comillas). Por ejemplo: SDR::MESDIF().

Sin embargo, esta forma de utilizar las funciones de nuestro paquete solo es práctica si estamos utilizándola en un paquete que estamos creando. Por lo que la forma más común de uso es enlazando el paquete al "search path" de R.

Para ello es necesario que primero esté instalado el paquete. Una vez instalado se enlaza utilizando la función `library("SDR")`.

AII.1.1. Funciones disponibles

Una vez cargado nuestro paquete y a modo de breve introducción mostramos las funciones que se pueden usar en el paquete. Si se desea mayor información acerca de parámetros y funcionalidad, por favor refiérase a la ayuda de la función utilizando `?funcion` (por ejemplo `?MESDIF`), o bien usando `help(funcion)`. Por ejemplo `help(MESDIF)`. Ahí encontrará ayuda muy detallada sobre la función:

- `SDIGA(…)`, `MESDIF(…)`, `NMEEF_SD(…)`: Ejecuta el algoritmo SDIGA, MESDIF o NMEEF-SD respectivamente. Los parámetros pueden ser introducidos a través de un fichero de parámetros o bien introducidos directamente en la función. Hay que destacar que la variable objetivo no se pasa como argumento, esto es

porque se asume que la última variable será siempre la variable objetivo. Para cambiar de variable se puede utilizar la función `changeTargetVariable()`.

- `read.keel(file, nLabels = 3)`: Lee un fichero de datos en formato de KEEL y crea el número de particiones difusas definidas para cada variable numérica.
- `changeTargetVariable(dataset, posVariable)`: Permite cambiar la variable objetivo de un dataset de KEEL ya cargado por `read.keel()` por otra distinta siempre y cuando esta sea categórica.
- `SDR_GUI()`: Lanza de manera local la interfaz web. Esta se abrirá en nuestro navegador predeterminado.

AII.1.2. Uso del paquete

En este apartado explicaremos como utilizar el paquete, haciendo uso para ello de un ejemplo.

Imaginemos que tenemos que utilizar MESDIF, para realizar una experimentación sobre el conjunto de datos "*german*". Este conjunto de datos se ha utilizado en esta memoria para realizar una de las experimentaciones comparativas entre algoritmos.

Tal experimentación, se puede llevar a cabo de dos manera diferentes, según como de preparado tengamos nuestros ficheros de datos:

- Tenemos en el conjunto de datos nuestra variable objetivo colocada como última variable del conjunto de datos.
- No tenemos en el conjunto de datos nuestra variable objetivo colocada como última variable del conjunto de datos.

En el primer caso, al tener el dataset preparado y no es necesario realizar ninguna modificación podremos realizar las dos formas de ejecución, a través de un fichero de parámetros o introduciendo manualmente los parámetros.

Si utilizamos un fichero de parámetros, este debe de tener un formato específico para cada algoritmo, en el caso de MESDIF, deberá ser el siguiente:

```
algorithm = MESDIF
inputData = "german-5-1tra.dat" "german-5-1tst.dat"
outputData = "german-5-1INFO.txt" "german-5-1-tra_Reg.txt" "german-5-1-tST_QM.txt"
seed = 0
nLabels = 3
nEval = 10000
popLength = 100
eliteLength = 3
crossProb = 0.6
mutProb = 0.01
RulesRep = can
echo = no
targetClass = null
Obj1 = CSUP
Obj2 = CCNF
Obj3 = null
Obj4 = null
```

Figura A II.1. Fichero de parámetros para ejecutar MESDIF

Si desea conocer los detalles del fichero de estos parámetros o los parámetros del resto de algoritmos, por favor, acuda a la ayuda de la función en la consola de R.

Una vez tengamos los parámetros preparados, será necesario antes de ejecutar el algoritmo desplazarse hasta el directorio donde se contengan los datos si, como en el ejemplo, no se ha especificado ruta completa de la ubicación del fichero. Esto se realiza utilizando la funcion `setwd()` e indicando la ruta a donde queremos ir.

Cuando estemos situados en la ruta, podremos ejecutar nuestro algoritmo. En nuestro caso, al tener el fichero de parámetros en el mismo directorio que nuestro conjunto de datos, no es necesario especificar ruta completa. Así pues, como nuestro fichero de parámetros se llama "**PARAM1.txt**" la orden para ejecutar MESDIF en este caso sería `MESDIF(paramFile = "PARAM1.txt")`.

```
>
>
>
>
>
> setwd("E:/Escritorio/webPlatform/webPlatform/data")
> MESDIF("PARAM1.txt")
```

Figura A II.2. Ejecución de MESDIF en R usando fichero de parámetros.

Si no tenemos el dataset preparado o queremos realizar la experimentación atendiendo a otra variable objetivo, debemos añadir los parámetros manualmente en nuestro algoritmo.

Antes de ejecutar la función, deberemos de cargar en memoria tanto el fichero de entrenamiento como el de test. Para cargarlo, debemos usar la función read.keel especificando la ruta del fichero y el número de etiquetas difusas que se usarán para cada variable. Si no especificamos este valor, se usarán tres etiquetas difusas por variable. En el ejemplo, usaremos cinco etiquetas difusas.

```
> setwd("E:/Escritorio/webPlatform/webPlatform/data")
> entrenamiento <- read.keel(file = "german-5-1tra.dat", nLabels = 5)
> test <- read.keel(file = "german-5-1tst.dat", nLabels = 5)
>
```

Figura A II.3. Carga en memoria de los conjuntos de entrenamiento y test.

Como se puede apreciar en la ilustración anterior, nos hemos desplazado al directorio donde tenemos el conjunto de datos para evitar usar rutas completas.

Una vez cargados los ficheros, si queremos cambiar de variable objetivo, deberemos utilizar en ambos ficheros la funcion changeTargetVariable(), la cual intercambia la variable que se encuentra en la posición indicada con la última siempre que esta sea una variable categórica. Para nuestra experimentación utilizaremos la variable *"ForeignWorker*" como variable objetivo. Si desea verificar que el tipo de dato es correcto puede visualizar el nombre de la variable y su tipo a través de los campos de la lista "*atributeNames*" y "*atributeTypes*" respectivamente.

Para utilizar "*ForeignWorker*" como variable objetivo, es necesario saber su posición. En el ejemplo de la ilustración, aparte de comprobar su tipo, comprobamos su posición. *"ForeignWorker"* ocupa la posición 20. Por lo tanto usaremos changeTargetVariable(dataset = entrenamiento, posVariable = 20)

```
> paste( seq_len(entramiento$nVars + 1),".-", entrenamiento$atributeNames, " -> ", entrenamiento$atributeTypes )
 [1] "1 .- StatusAccount  ->  c"     "2 .- DurationMonth  ->  e"     "3 .- CreditHistory  ->  c"
 [4] "4 .- Purpose  ->  c"           "5 .- CreditAmount  ->  e"      "6 .- SavingsAccount  ->  c"
 [7] "7 .- EmploymentSince  ->  c"   "8 .- InstallmentRate  ->  e"   "9 .- StatusAndSex  ->  c"
[10] "10 .- Guarantors  ->  c"       "11 .- ResidenceSince  ->  e"   "12 .- Property  ->  c"
[13] "13 .- Age  ->  e"              "14 .- InstallmentPlans  ->  c" "15 .- Housing  ->  c"
[16] "16 .- NCredits  ->  e"         "17 .- Job  ->  c"              "18 .- NPeopleMain  ->  e"
[19] "19 .- Telephone  ->  c"        "20 .- ForeignWorker  ->  c"    "21 .- Customer  ->  c"
> entrenamiento <- changeTargetVariable(dataset = entrenamiento, posVariable = 20)
> test <- changeTargetVariable(dataset = test, posVariable = 20)
```

Figura A II.4. Cambio de variable objetivo desde la consola de R

En el ejemplo de la Figura II.4 se muestra como hemos comprobado tipo y posición de las variables.

Ahora ya estamos en disposición de lanzar nuestro algoritmo, usando la ejecución con los mismos parámetros que se especificaron anteriormente en el fichero de parámetros, la llamada a la función quedaría como se muestra en la siguiente imagen:

```
> paste( seq_len(entramiento$nVars + 1),".-", entrenamiento$atributeNames, " -> ", entrenamiento$atributeTypes )
 [1] "1 .- StatusAccount  ->  c"     "2 .- DurationMonth  ->  e"     "3 .- CreditHistory  ->  c"
 [4] "4 .- Purpose  ->  c"           "5 .- CreditAmount  ->  e"      "6 .- SavingsAccount  ->  c"
 [7] "7 .- EmploymentSince  ->  c"   "8 .- InstallmentRate  ->  e"   "9 .- StatusAndSex  ->  c"
[10] "10 .- Guarantors  ->  c"       "11 .- ResidenceSince  ->  e"   "12 .- Property  ->  c"
[13] "13 .- Age  ->  e"              "14 .- InstallmentPlans  ->  c" "15 .- Housing  ->  c"
[16] "16 .- NCredits  ->  e"         "17 .- Job  ->  c"              "18 .- NPeopleMain  ->  e"
[19] "19 .- Telephone  ->  c"        "20 .- ForeignWorker  ->  c"    "21 .- Customer  ->  c"
> entrenamiento <- changeTargetVariable(dataset = entrenamiento, posVariable = 20)
> test <- changeTargetVariable(dataset = test, posVariable = 20)
>
>
>
> MESDIF(training = entrenamiento,
         test = test,
         output = c("german-5-1INFO.txt","german-51-tra_Reg.txt", "german-5-1-tST_QM.txt"),
         seed = 0,
         nLabels = 5,
         nEval = 10000,
         popLength = 100,
         eliteLength = 3,
         crossProb = 0.6,
         mutProb = 0.01,
         RulesRep = "can",
         Obj1 = "CSUP",
         Obj2 = "CCNF",
         Obj3 = "null",
         Obj4 = "null",
         targetClass = "null"
  )
```

Figura A II.5. Ejecución de MESDIF a partir del uso de los parámetros de la función

El proceso es similar para el resto de algoritmos, lo único que cambia son sus parámetros y la llamada a la función.

Cuando ejecutemos el algoritmo, los resultados se guardarán en los ficheros especificados en el parámetro output, cuyo orden es: Información de la ejecución, fichero de reglas generas y fichero de medidas de calidad. Asimismo se nos mostrarán por pantalla (dependiendo del fichero de datos y de las reglas que se generen, esta salida puede ser muy grande) los mismos resultados que se guardarán en los ficheros de resultados. Este resultado se divide de la siguiente manera:

- Información de la ejecución: Una vez ejecutemos el algoritmo, se nos mostrará a modo informativo, los parámetros utilizados así como información de los conjuntos de datos.

```
>
> MESDIF(training = entrenamiento,
+        test = test,
+        output = c("german-5-1INFO.txt","german-51-tra_Reg.txt", "german-5-1-tST_QM.txt"),
+        seed = 0,
+        nLabels = 5,
+        nEval = 10000,
+        popLength = 100,
+        eliteLength = 3,
+        crossProb = 0.6,
+        mutProb = 0.01,
+        RulesRep = "can",
+        Obj1 = "CSUP",
+        Obj2 = "CCNF",
+        Obj3 = "null",
+        Obj4 = "null",
+        targetClass = "null"
+ )
--------------------------------
Algorithm: MESDIF
Relation: german
Training file:
Test file:
Rules Representation:  CAN
Number of evaluations: 10000
Number of fuzzy partitions: 5
Population Length: 100
Elite Population Length: 3
Crossover Probability: 0.6
Mutation Probability: 0.01
Obj1: CSUP   (Weigth: )
Obj2: CCNF   (Weigth: )
Obj3: null   (Weigth: )
Obj4: null
Number of examples in training: 800
Number of examples in test: 200
--------------------------------
```

Figura A II.6. Información sobre la ejecución del algoritmo.

- Justo debajo de este apartado, aparecen las reglas generadas. En este ejemplo se ha realizado una búsqueda para todos los valores de la variable objetivo (usando el valor "*null*" en *"targetClass"*)

```
 Searching rules for all values of the target class...


 ? Target value: A201

GENERATED RULE 1
Variable Guarantors = A101
 THEN A201



GENERATED RULE 2
Variable InstallmentRate = Label 4 ( 3.25 , 4 , 4.75 )
Variable Telephone = A192
 THEN A201



GENERATED RULE 3
Variable Guarantors = A101
Variable Job = A173
 THEN A201



 ? Target value: A202

GENERATED RULE 1
Variable CreditAmount = Label 0 ( -4293.5 , 250 , 4793.5 )
Variable SavingsAccount = A64
Variable ResidenceSince = Label 1 ( 1 , 1.75 , 2.5 )
Variable Housing = A152
Variable NCredits = Label 0 ( 0.25 , 1 , 1.75 )
Variable Telephone = A191

 THEN A202
```

Figura A II.7. Reglas generadas por el algoritmo.

- A continuación de las reglas se mostrarán las medidas de calidad obtenidas por las reglas en el conjunto de test.

```
 Testing rules...


Rule 1 :
        - N_vars: 2
        - Coverage: 0.915
        - Significance: 0.114516
        - Unusualness: 0.075225
        - Accuracy: 0.962162
        - CSupport: 0.885
        - FSupport: 0.885
        - CConfidence: 0.967213
        - FConfidence: 0.967213


Rule 2 :
        - N_vars: 3
        - Coverage: 0.2
        - Significance: 1.418301
        - Unusualness: 0.16
        - Accuracy: 0.97619
        - CSupport: 0.2
        - FSupport: 0.2
        - CConfidence: 1
        - FConfidence: 1


Rule 3 :
        - N_vars: 3
        - Coverage: 0.545
        - Significance: 0.718988
        - Unusualness: 0.243425
        - Accuracy: 0.972973
        - CSupport: 0.535
        - FSupport: 0.535
        - CConfidence: 0.981651
        - FConfidence: 0.981651
```

Figura A II.8. Medidas de calidad de las reglas aplicadas al conjunto de test.

El ultimo valor de test ("*Global*"), muestra un resumen de las reglas obtenidas, mostrando el valor medio de las medidas de calidad de todas las reglas.

En los ficheros de resultados el resultado obtenido es idéntico, sin embargo, cada uno de los tres puntos explicados anteriormente está guardado en un fichero diferente.

Manual de uso de la interfaz

Utilizar la plataforma web es posible desde dos sitios distintos:

- Usando la función SDR_GUI(), que lanza la interfaz de manera local.
- Visitando http://sdrinterface.shinyapps.io/shiny

AII.1.3. Análisis exploratorio de datos

Independientemente del lugar desde donde accedamos a la interfaz, esta se encuentra inicialmente como se muestra en la siguiente Ilustración:

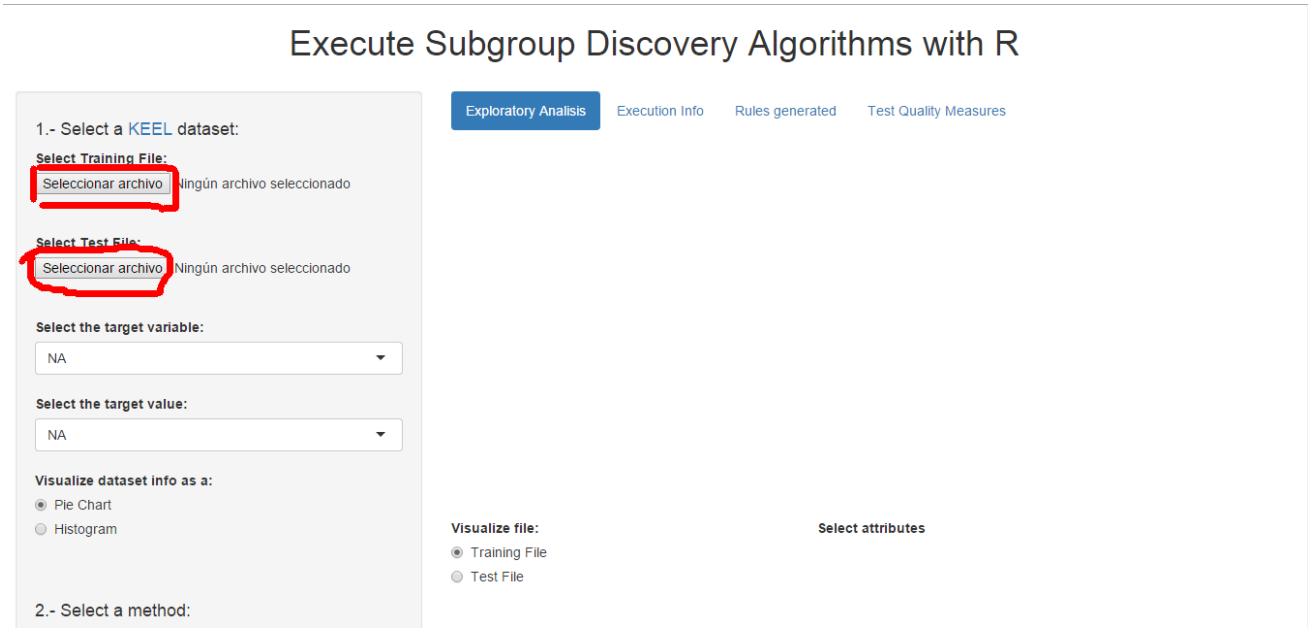


Figura A II.10. Interfaz en su estado inicial.

En rojo, se ha señalado la única interacción posible, que es cargar un conjunto de entrenamiento, o uno de prueba en formato de KEEL. Una vez seleccionado nuestro fichero, el resultado que se nos muestra será el siguiente (se ha utilizado como conjunto de ejemplo el *dataset* "*german*").

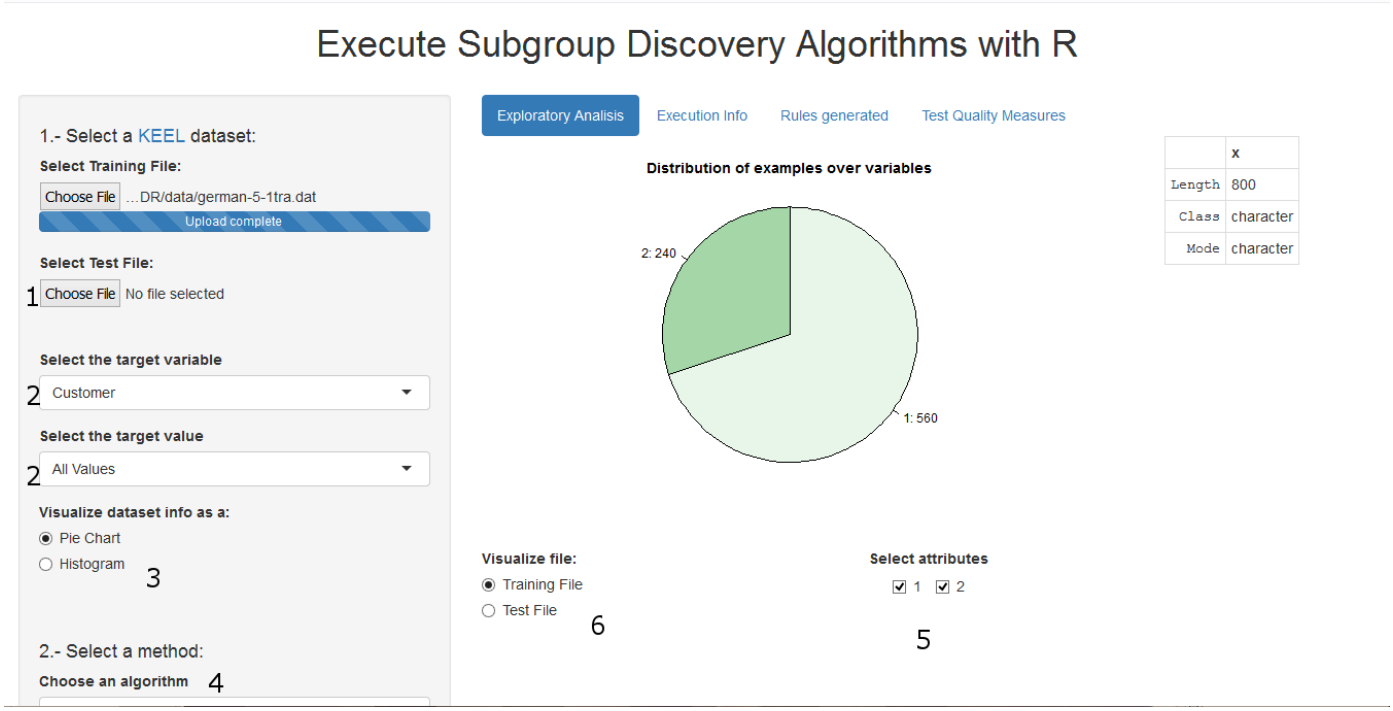


Figura A II.9. Interfaz con un dataset cargado.

Una vez seleccionado uno de los dos ficheros necesarios para ejecutar nuestros algoritmos, se nos abre todo un abanico de posibilidades a realizar para el análisis exploratorio de nuestro conjunto de datos. Cada una de las acciones que podemos realizar se encuentran marcadas en la figura por un número.

- El numero uno, permite cargar otro dataset. Una vez cargados los dos, si son de la misma relación (comparten atributo *@relation*) es posible lanzar los algoritmos si lo deseamos.
- Los numeros dos tienen una doble función. Por un lado, permiten seleccionar la variable objetivo del algoritmo, así como el valor objetivo, y por otro muestran el gráfico de la variable seleccionada.
- El numero tres permite el cambio de visualización de una gráfica de sectores a un histograma. Las acciones de los puntos 2 y 3 se recogen en la siguiente figura:

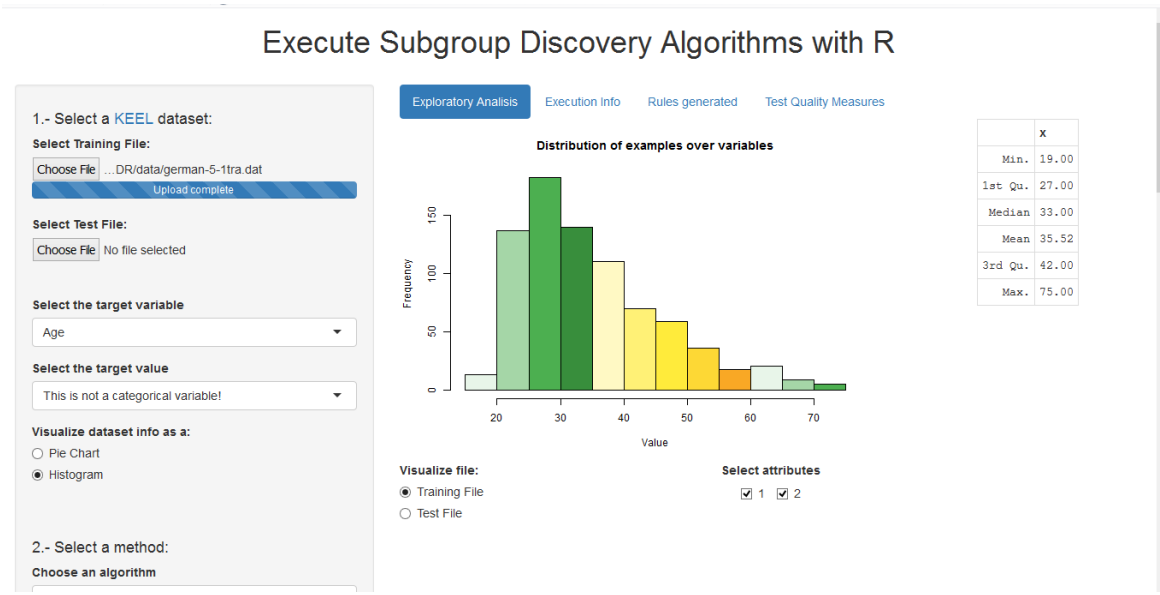


Figura A II.11. Visualización de los datos en forma de histograma.

- El número cuatro permite seleccionar el algoritmo y modificar sus parámetros. Hablaremos de él mas adelante
- El número cinco permite seleccionar, para variables categóricas, los valores de dicha variable que deberán mostrarse en la gráfica. Hay que destacar que lo único que se hace es quitar de la gráfica aquellos valores que no se encuentren marcados, así que los ejemplos que contengan los valores desmarcados seguiran siendo “visibles” para el algoritmo.

- El número seis permite cambiar la visualización para el conjunto de entrenamiento o para el de prueba.

AII.1.4. Ejecución de los algoritmos

Una vez hayamos visualizado el conjunto de datos, el siguiente paso lógico es ejecutar un algoritmo de descubrimiento de subgrupos. La selección de un algoritmo, así como la modificación de sus parámetros, se muestra tal y como aparece en la Figura A II.12:

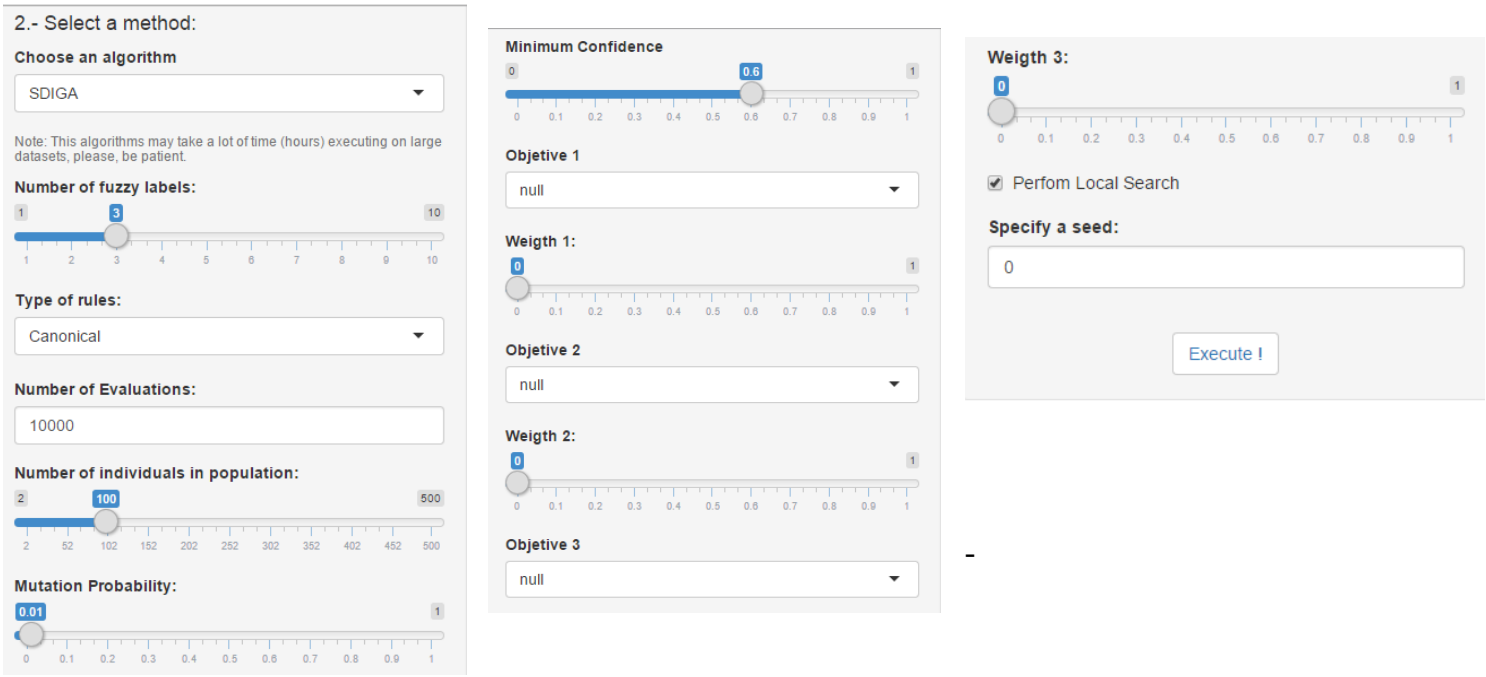


Figura A II.12. Parámetros modificables desde la interfaz para el algoritmo SDIGA

Las imágenes se muestran de izquierda a derecha en el orden en que saldrían en vertical. Como se puede apreciar, se pueden modificar todos los parámetros de cada algoritmo.

Cuando pulsemos en ejecutar, el algoritmo comenzará a ejecutarse buscando subgrupos para la variable objetivo seleccionada y para el valor objetivo que tuviéramos seleccionado. Una vez finalice el algoritmo se mostrarán los resultados obtenidos en cada una de las diferentes pestañas:

- La pestaña "*Rules generated*" nos muestra las reglas que se han obtenido. Esta pestaña se activa siempre que finaliza la ejecución de un algoritmo.

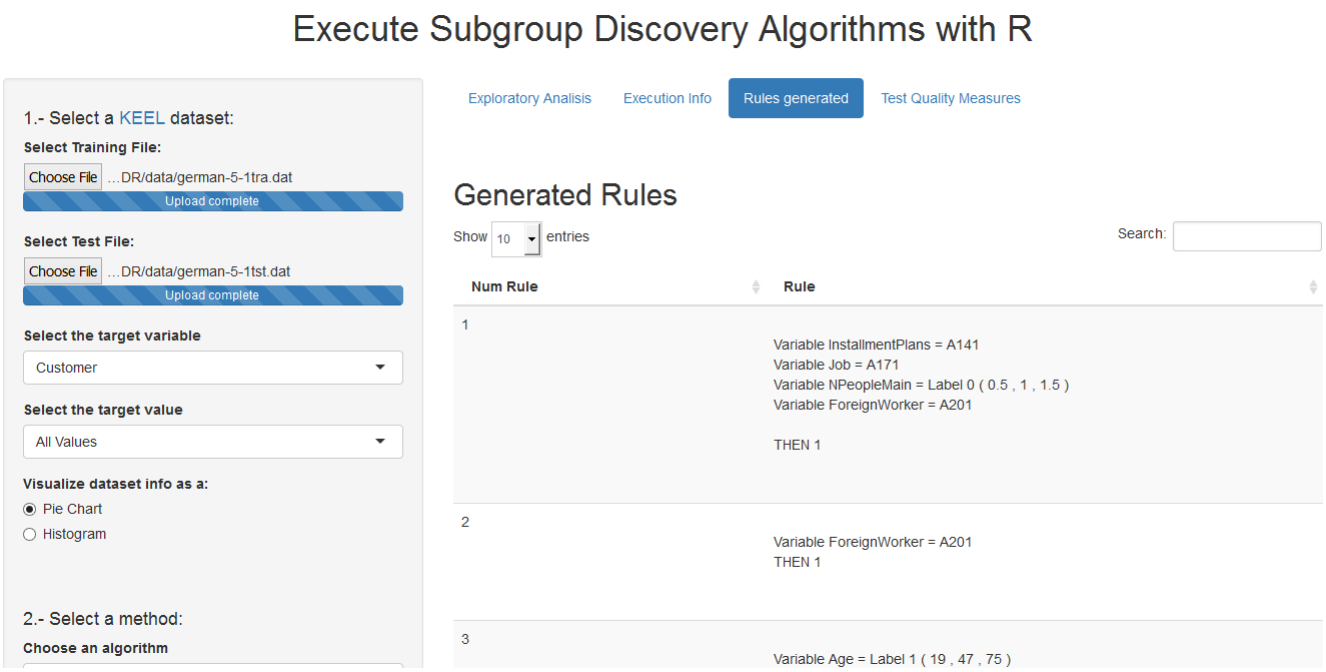


Figura A II.13. Visualización de las reglas obtenidas.

- La pestaña "*Test quality measures"* muestras las medidas de calidad de cada regla aplicadas al conjunto de pruebas.

- La pestaña "*Execution Info*" nos muestra información sobre la ejecución llevada a cabo, es decir, nos muestra los parámetros que se han utilizado para llevar a cabo esa experimentación.