Usando Algoritmos de Descubrimiento de Subgrupos en R: El Paquete SDR

Ángel M. García[1], Francisco Charte[2], Pedro González[1], Cristóbal J. Carmona[3], and María J. del Jesus[1]

[1] Departamento de Informática, Universidad de Jaén
amgv0009@red.ujaen.es
[2] Departamento de Ciencias de la Computación e Inteligencia Artificial, Universidad de Granada.
[3] Área de Lenguajes y Sistemas del Departamento de Ingeniería Civil, Universidad de Burgos

Resumen El descubrimiento de subgrupos es una tarea de la minería de datos entre la clasificación y la descripción. Esta tarea es de gran interés para los investigadores debido a su éxito en campos como Medicina o Bioinformática. En este trabajo se describe un paquete de algoritmos para la extracción de reglas de descripción de subgrupos desarrollado íntegramente en R y disponible en CRAN. El paquete incorpora una interfaz gráfica que permite el uso sencillo de los algoritmos, sin necesidad de ejecutar directamente los comandos en la consola de R. Dicha interfaz además permite realizar un análisis exploratorio de datos básico.

Keywords: Descubrimiento de subgrupos, R, Reglas descriptivas, algoritmos evolutivos

1. Introducción

En casi cualquier actividad cotidiana se generan grandes cantidades de información. Esta procede de Internet, comunicaciones, redes de sensores, etc. Dicha información contiene conocimiento implícito que es muy importante para las organizaciones para, por ejemplo, mejorar sus productos o servicios [1]. Para conseguir este propósito, se utiliza la minería de datos donde existen dos enfoques clásicos: un enfoque supervisado, que normalmente tiene un comportamiento predictivo y un enfoque no supervisado, cuyo comportamiento es normalmente descriptivo. Sin embargo, existen algunas técnicas de minería de datos que se encuentran entre estos dos enfoques, como el descubrimiento de subgrupos (SD) [2]. En este trabajo nos centraremos en el descubrimiento de subgrupos, que permite obtener reglas "interesantes" para el usuario, buscando para ello reglas simples, muy generales, precisas y con gran interés.
Actualmente R [3] es una de las herramientas más utilizadas en ciencia de datos [4] y, sin embargo, en su repositorio de paquetes más importante, *CRAN (the Comprenhensive R Archive Network)* no existía ningún paquete con algoritmos

de SD implementados íntegramente en R, por lo que el paquete cubre esta necesidad. Asimismo, el paquete propuesto posee una interfaz gráfica para facilitar el uso del paquete al no tener que escribir los comandos directamente en la consola de R.
Este trabajo se estructura de la siguiente forma: en la Sección 2 se introduce brevemente la tarea descubrimiento de subgrupos. En la Sección 3 se describen brevemente los algoritmos existentes en el paquete propuesto así como sus características más destacables. En la Sección 4 se explica el uso del paquete propuesto desde la consola de R. En la Sección 5 se explica la interfaz gráfica que contiene el paquete y por último, la Sección 6 detalla las conclusiones y el trabajo futuro.

2. Descubrimiento de Subgrupos

El descubrimiento de subgrupos (SD) se puede definir como en [5]:

> "Partiendo de una población de individuos y de una propiedad de interés, la tarea de descripción de subgrupos puede ser definida como la obtención de subgrupos de individuos de la población con propiedades estadísticamente «interesantes», es decir, lo más grandes posibles y con una distribución estadística inusual respecto a la propiedad de interés."

El objetivo de cualquier algoritmo de SD es, por tanto, obtener propiedades interesantes de estos subgrupos. Dichas propiedades se representan con reglas del tipo:

$$R : Cond \rightarrow Clase \tag{1}$$

Donde $Clase$ es nuestra variable de interés y $Cond$ es un conjunto de características del subgrupo obtenido, representadas como conjunciones de pares atributo-valor o bien en forma normal disyuntiva.

2.1. Medidas de Calidad para Descubrimiento de Subgrupos

Las medidas de calidad son muy importantes en el descubrimiento de subgrupos para la obtención de reglas precisas, simples e interpretables. A día de hoy no existe un consenso sobre qué medida es mejor usar. Las más utilizadas en la bibliografía especializada y consideradas en este trabajo son [6]:

- $\boldsymbol{N_r}$: Número de reglas generadas.
- $\boldsymbol{N_v}$: Número medio de variables que poseen las reglas generadas.
- **Soporte:** Mide la frecuencia de ejemplos correctamente clasificados por la regla:

$$Sup(R) = \frac{n(Cond \wedge Clase)}{n_s} \tag{2}$$

donde R es la regla a evaluar, $n(Cond \wedge Clase)$ indica el número de ejemplos que verifican el antecedente y el consecuente de la regla y n_s el número de ejemplos del *dataset*.

- **Cobertura:** Mide el porcentaje de ejemplos cubiertos por la regla respecto al total de ejemplos:

$$Cov(R) = \frac{n(Cond)}{n_s} \tag{3}$$

donde $n(Cond)$ indica el número de ejemplos que cumplen el antecedente de la regla.

- **Confianza:** Mide el porcentaje de ejemplos correctamente clasificados del total de ejemplos cubiertos:

$$Conf(R) = \frac{n(Cond \wedge Clase)}{n(Cond)} \tag{4}$$

- **Significancia:** Refleja la novedad en la distribución de los ejemplos cubiertos por la regla respecto al conjunto de ejemplos completo:

$$Sign(R) = 2 \cdot \sum_{k=1}^{n_c} n(Cond \wedge Clase_k) \cdot log\left(\frac{n(Cond \wedge Clase_k)}{n(Cond \wedge Clase) \cdot p(Cond)}\right) \tag{5}$$

donde $p(Cond) = \frac{n(Cond)}{n_s}$, n_c indica el número de valores que posee la variable objetivo y $Clase_k$ indica el valor k-ésimo de la variable objetivo.

- **Atipicidad:** Se define como la precisión ponderada relativa de la regla:

$$WRAcc(R) = \frac{n(Cond)}{n_s} \cdot \left(\frac{n(Cond \wedge Clase)}{n(Cond)} - \frac{n(Clase)}{n_s}\right) \tag{6}$$

donde $n\,(Clase)$ es el número de ejemplos que pertenecen a la variable objetivo.

3. Algoritmos Implementados

Los algoritmos que actualmente se encuentran implementados en el paquete propuesto son: SDIGA [7], MESDIF [8] y NMEEF-SD [9]. Estos algoritmos comparten las siguientes características principales:

- Utilizan un enfoque evolutivo como estrategia de búsqueda. Este enfoque evolutivo se implementa como un algoritmo genético mono-objetivo [10] para SDIGA y multi-objetivo [11] para MESDIF y NMEEF-SD.
- Hacen uso de lógica difusa [12] para tratar con datos numéricos, aumentando la interpretabilidad de los resultados obtenidos.
- Permiten una representación del antecedente de las reglas como canónicas (pares atributo-valor) o bien en forma normal disyuntiva (DNF).
- Permiten especificar las medidas de calidad a utilizar para definir los subgrupos. Estas medidas son las enumeradas en la Sección 2.1.

Asimismo, las limitaciones temporales de estos algoritmos vienen determinadas por dos factores: la dimensionalidad del conjunto de datos y el tamaño de la población o número de reglas que se tratarán en el proceso evolutivo.

- Para SDIGA, la complejidad de su algoritmo genético es $O(R \cdot N \cdot V)$ siendo R el número de reglas, N el número de ejemplos y V el número de variables, aunque normalmente $R << N$ y $R << V$ por lo que en la práctica la complejidad es $O(N \cdot V)$. Por otra parte, la complejidad del algoritmo de optimización local es de $O(N \cdot V^{(V-1)})$, pero este caso es muy poco probable debido al proceso genético previo y normalmente el caso medio se encuentra en $\Theta(N \cdot V)$. Por lo tanto, podemos afirmar que el algoritmo tiene un orden de complejidad de $O(N \cdot V)$, es decir, lineal respecto al tamaño del conjunto de entrenamiento.
- Para MESDIF y NMEEF-SD, evaluar la población también tiene un coste de $O(N \cdot V)$, sin embargo hay que destacar que su proceso evolutivo tiene un coste de $O(m \cdot R^2)$ siendo m el número de medidas de calidad a usar. Puesto que, en general $R << N$ y $R << V$ la complejidad computacional sería equivalente a la del caso previo.

4. Uso del Paquete SDR

En este apartado describiremos cómo usar el paquete propuesto. Su finalidad es hacer que el uso de los algoritmos de descubrimiento de subgrupos incluidos sea muy sencillo.

4.1. Instalación del Paquete

Para instalar el paquete en R, una vez abierta la consola simplemente escribiremos:

```
> install.packages("SDR")
```

Asimismo, la versión en desarrollo se encuentra disponible en `http://github.com/aklxao2/SDR` , si desea instalar la versión en desarrollo, debe de instalar el paquete `devtools` y a continuación ejecutar la siguiente orden:

```
> devtools::install_github("aklxao2/SDR")
```

El paquete SDR depende únicamente del paquete `shiny` [13]. Este paquete será necesario para la utilización de la interfaz gráfica. Una vez instalado el paquete es necesario cargarlo antes de poder utilizarlo, para cargarlo se debe ejecutar la siguiente orden:

```
> library(SDR)
```

o bien

```
> require(SDR)
```

Una vez cargado el paquete se encuentran disponibles seis conjuntos de datos: `habermanTra`, `habermanTst`, `carTra`, `carTst`, `germanTra` y `germanTst` correspondientes a los conjuntos de datos de entrenamiento y prueba de los *datasets* `car` [14], `haberman` [15] y `german` [16] disponibles además en el repositorio de conjuntos de datos de KEEL [17].

4.2. Carga de un *Dataset* de *KEEL*

Es posible cargar en R un fichero de datos en el formato de la herramienta de minería de datos KEEL [18]. Teniendo los archivos `irisTra.dat` y `irisTst.dat` en el directorio de trabajo de R, estos se cargarán mediante la función `read.keel()`:

```
> irisTra <- read.keel("irisTra.dat")
> irisTst <- read.keel("irisTst.dat")
```

Como se ha comentado en la Sección 3 los algoritmos implementados usan lógica difusa. Por defecto, la función `read.keel()` define tres particiones difusas triangulares [9, pag. 4] de igual anchura para cada una de las variables numéricas que existan en el conjunto de datos. Si se desea especificar un número diferente de particiones difusas a la hora de cargar el fichero en R podemos realizarlo especificando el parámetro `nLabels`:

```
> irisTra <- read.keel("irisTra.dat", nLabels = 5)
> irisTst <- read.keel("irisTst.dat", nLabels = 5)
```

En este caso, se cargan los ficheros de datos con 5 etiquetas difusas para cada variable numérica.

4.3. Cambio de Propiedades de un *Dataset* KEEL Cargado

Una vez cargado un fichero, se pueden realizar una serie de modificaciones en su estructura. Estas modificaciones son:

- Cambiar la variable objetivo usada. Por defecto, los algoritmos implementados utilizan como variable objetivo la última variable que se encuentra en el *dataset*. Esta cambio se realiza a través de la instrucción `changeTargetVariable`.

  ```
  > irisTra <- changeTargetVariable(irisTra, 2)
  ```

  En este caso, se especifica como variable objetivo la variable que ocupa la posición dos en la definición de variables del *dataset*. Para que una variable pueda utilizarse como variable objetivo, esta debe ser categórica.
- Cambiar la cantidad de etiquetas difusas que se definen para las variables numéricas del *dataset*.

  ```
  > irisTra <- modifyFuzzyCrispIntervals(irisTra, 7)
  ```

  En este caso, se utilizarán siete particiones difusas para las variables numéricas, nótese que anteriormente se especificaron cinco particiones difusas.

4.4. Ejecución de los Algoritmos de Descubrimiento de Subgrupos

Una vez configurado el conjunto de datos es posible ejecutar uno de los algoritmos de descubrimiento de subgrupos. Para cada uno de los algoritmos se encuentra disponible una función. Estas funciones son muy similares entre sí, cambiando ligeramente los parámetros a usar. Para más información, utilice la ayuda de la función mediante `?funcion` o `help(funcion)` en la consola de R. Un algoritmo de descubrimiento de subgrupos implementado en el paquete propuesto puede ser utilizado de dos maneras distintas:

- A través de un fichero de parámetros: en este caso se debe indicar únicamente la ruta hacia dicho fichero. Su uso es indicado para conjuntos de datos que se tienen preparados de antemano, pues no se permite modificar estos *datasets* antes de la ejecución del algoritmo. Suponiendo el fichero de parámetros `ParamFile.txt` en el directorio de trabajo de R, un ejemplo de ejecución sería:

```
> MESDIF("ParamFile.txt")
```

- Escribiendo en la consola todos y cada uno de los parámetros necesarios para la ejecución. Este método es el indicado si ha cargado y configurado un *dataset* previamente. Supongamos la utilización del algoritmo anterior con el conjunto `haberman` que se encuentra disponible una vez se carga el paquete:

```
> MESDIF( paramFile = NULL,
training = habermanTra,
test = habermanTst,
output = c("optionsFile.txt", "rulesFile.txt", "testQM.txt"),
seed = 0,
nLabels = 3,
nEval = 300,
popLength = 100,
eliteLength = 3,
crossProb = 0.6,
mutProb = 0.01,
RulesRep = "can",
Obj1 = "CSUP",
Obj2 = "CCNF",
Obj3 = "null",
Obj4 = "null",
targetClass = "positive"
)
```

Una vez ejecutado, los resultados se mostrarán en la consola de R, divididos en tres secciones claramente diferenciadas:

- Primero, un resumen de los parámetros usados en la ejecución del algoritmo.
- Segundo, se muestran las reglas o subgrupos obtenidos por el algoritmo.

- Finalmente, se muestran para cada regla obtenida algunas medidas de calidad aplicando dichas reglas al conjunto de datos de prueba. El valor final es un resumen de las medidas de calidad para todas las reglas.

5. La Interfaz de Usuario

Como hemos mencionado al inicio de este trabajo, el paquete SDR proporciona una interfaz gráfica que facilita el uso y la experiencia de usuario con el paquete propuesto así como realizar tareas básicas de análisis exploratorio de datos. Podemos utilizar la interfaz de dos formas: visitando `http://sdrinterface.shinyapps.io/shiny` o bien lanzando dicha interfaz de manera local mediante la función:

```
> SDR_GUI()
```

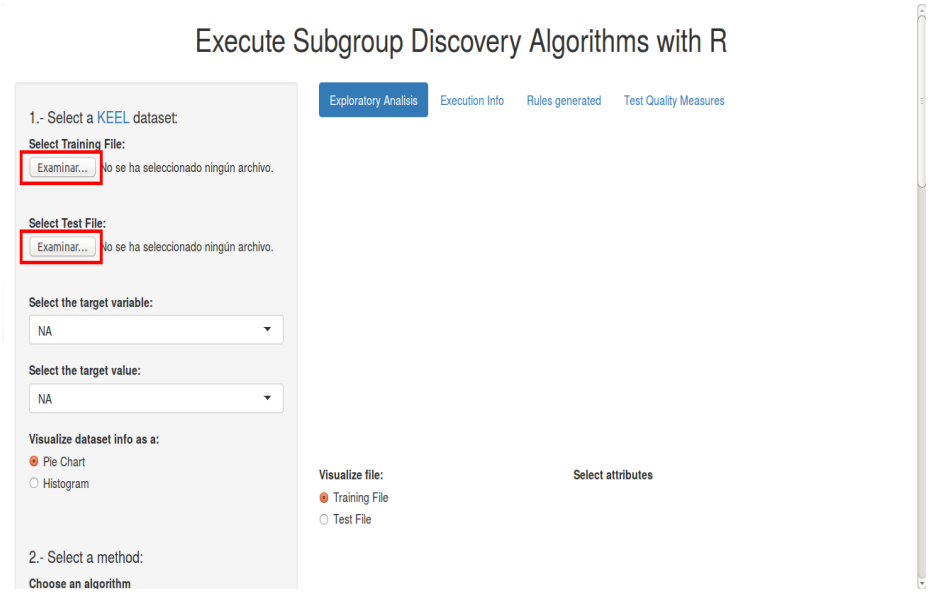

Figura 1. Pantalla inicial de la interfaz gráfica de SDR.

Esta función lanza la interfaz de usuario en el navegador web predeterminado. Como podemos ver en la figura 1 la página está organizada en un panel de opciones a la izquierda y un panel de pestañas a la derecha, donde se mostrarán los resultados. El primer paso que se debe realizar es cargar un conjunto de datos en formato KEEL. Si se desea ejecutar algún algoritmo de descubrimiento de subgrupos, ambos ficheros (entrenamiento y prueba) deben tener el mismo campo `@relation`. Una vez seleccionado un conjunto de datos, se muestra un

gráfico con información sobre las variables. Por defecto, se muestra la distribución de ejemplos sobre los diferentes valores de la variable seleccionada, que es la última del *dataset*. Asimismo, se muestra a la derecha de este gráfico una tabla con medidas estadísticas comunes.

Una vez cargado un conjunto de datos se abre un abanico de acciones a realizar. En la figura 2 pueden verse todas las opciones disponibles (marcadas con números del 1 al 6):

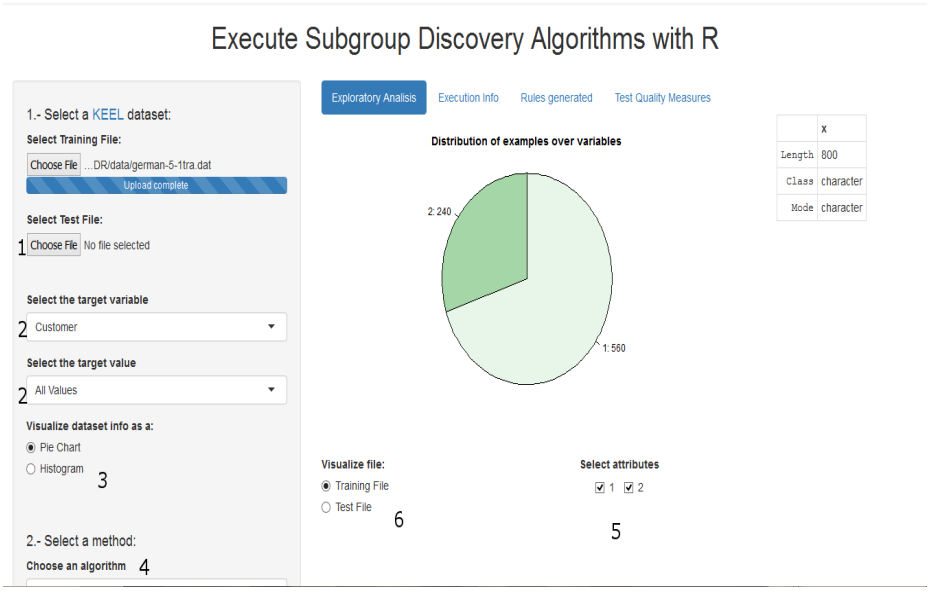

Figura 2. Interfaz una vez cargado un conjunto de datos.

1. Como se ha mencionado anteriormente, se puede cargar otro conjunto de datos de prueba (o entrenamiento).
2. Estos menús desplegables tienen una doble función: por un lado permiten seleccionar la variable a visualizar en el gráfico y por otro definen la variable objetivo en el conjunto de datos. El segundo menú solo sirve para indicar al algoritmo de descubrimiento de subgrupos el valor de dicha variable o bien buscar subgrupos para todos los posibles valores de la variable objetivo.
3. Aquí se puede elegir la visualización de la información del conjunto de datos, para variables categóricas podemos elegir entre un diagrama de sectores o bien un histograma. Para variables numéricas solo se puede usar un histograma.
4. En esta sección se puede elegir el algoritmo de descubrimiento de subgrupos a ejecutar así como modificar sus parámetros. Debajo de todos los parámetros, se encuentra el botón que inicia la ejecución del algoritmo de descubrimiento de subgrupos.

5. Este apartado permite la selección, para variables categóricas, de los valores que se visualizarán en la gráfica. Es importante remarcar que solo "oculta" los valores, por lo que no elimina ningún ejemplo del *dataset*.
6. Permite visualizar la información del conjunto de entrenamiento o de conjunto de prueba.

Una vez ejecutado un algoritmo, automáticamente se pasa a la pestaña `Rules generated` donde se muestran las reglas obtenidas. Si se desea filtrar variables, por ejemplo aquellas que contengan una variable en concreto, es posible hacerlo escribiendo en el campo de búsqueda.
La pestaña `Execution Info` muestra un eco de los paramétros de ejecución. Este eco es igual que el que se muestra en la consola de R.
La pestaña `Test Quality Measures` muestra una tabla con las medidas de cada regla aplicada al conjunto de prueba. La última fila de esta tabla muestra un resumen con el valor medio de cada medida de calidad.

6. Conclusiones

En este artículo se ha presentado el paquete SDR, que contiene actualmente tres algoritmos evolutivos de descubrimiento de subgrupos que no dependen de ningún paquete/herramienta adicional y/o externa. También se ofrece la posibilidad de leer y cargar un conjunto de datos en el formato que proporciona la herramienta de minería de datos KEEL. Asimismo posee una serie de herramientas para hacer pequeñas modificaciones a estos conjuntos de datos cargados en memoria. Para finalizar, una interfaz gráfica basada en web ha sido desarrollada para facilitar al usuario el uso del paquete propuesto, incluso si no tiene R instalado.
El desarrollo de este paquete continuará en el futuro, incluyendo mayor funcionalidad para el tratamiento de conjuntos de datos de KEEL, mayor variedad de algoritmos de descubrimientos de subgrupos así como mejoras en la interfaz gráfica como la adición de más cantidad de gráficas para ampliar las posibilidades de análisis exploratorio.
Agradecimientos: Este trabajo está parcialmente financiado por el proyecto TIN2012-33856.

Referencias

1. C.J. Carmona, S. Ramirez-Gallego, F. Torres, E. Bernal, M.J. del Jesus y S. Garcia: Web usage mining to improve the design of an e-commerce website: OrOliveSur.com. In Expert Systems with Applications, pp 11243-11249, (2012).
2. S. Wrobel: An algorithm for multi-relational discovery of subgroups. In: Proceeding of the 1st European symposium on principles of data mining and knowledge discovery, vol. 1263, pp 78-87. (1997).

3. R Core Team: R: A Laguage and Environment for Statistical Computing, Organization: R Foundation for Statistical Computing , Vienna, Austria, URL: http://www.R-project.org/, (2015).
4. Languages for analytics/data mining. [En línea]. Disponible en: http://www.kdnuggets.com/polls/2015/analytics-data-mining-data-science-software-used.html Visitada 19/06/2015.
5. S. Wrobel: Inductive logic programming for knowledge discovery in databases. In Relational Data Mining, pp 74-101. (2001).
6. F. Herrera, M. J. del Jesus, P. Gonzalez, C. J. Carmona: An overview on subgroup discovery: foundations and applications. In Knowledge and Information Systems. vol. 29, no. 3, pp 495-525. (2011).
7. M. J. del Jesus, P. Gonzalez, F. Herrera, M. Mesonero: Evolutionary fuzzy rule induction process for subgroup discovery: a case study in marketing. In IEEE Trans Fuzzy Syst. vol. 15, no. 4, pp 578-592, (2007).
8. M. J. del Jesus, P. Gonzalez, F. Herrera: Multiobjective genetic algorithm for extracting subgroup discovery fuzzy rules. In Proceedings of the IEEE symposium on computational intelligence in multi-criteria decision making. pp 50-57, (2007).
9. C. J. Carmona, P. Gonzalez, M. J. del Jesus, F. Herrera: NMEEF-SD: Non-dominated Multi-objective Evolutionary algorithm for Extracting Fuzzy rules in Subgroup Discovery. In Fuzzy Systems, IEEE Transactions. vol. 18, no. 5, pp 958-970, (2010).
10. T. Bäck, D. Fogel y Z. Michalewicz: Handbook of evolutionary computation. Oxford: Oxford University Press. (1997).
11. K. Deb: Multi-Objective Optimization using Evolutionary Algorithms. John Wiley & Sons, (2011)
12. H. Mamdani, S. Assilian: An experiment in linguistic synthesis with a fuzzy logic controller. In International journal of man-machine studies. vol. 7, no. 1, pp 1-13. (1975).
13. W. Chang: shiny: Web Application Framework for R, URL http://CRAN.R-project.org/package=shiny. R package version 0.11. (2015).
14. M. Bohanec and V. Rajkovic: Knowledge acquisition and explanation for multi-attribute decision making. In 8th Intl Workshop on Expert Systems and their Applications, Avignon, France. pp 59-78, (1988).
15. S. J. Haberman: Generalized Residuals for Log-Linear Models, Proceedings of the 9th International Biometrics Conference, Boston. pp 104-122. (1976).
16. German Credit Dataset. Disponible en: http://sci2s.ugr.es/keel/dataset.php?cod=88 Visitada 18/06/2015
17. J. Alcalá-Fdez, A. Fernandez, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera: KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework. In Journal of Multiple-Valued Logic and Soft Computing vol. 17 no. 2-3 pp 255-287. (2011)
18. J. Alcalá-Fdez, L. Sánchez, S. García, M.J. del Jesus, S. Ventura, J.M. Garrell, J. Otero, C. Romero, J. Bacardit, V.M. Rivas, J.C. Fernández, F. Herrera: KEEL: A Software Tool to Assess Evolutionary Algorithms to Data Mining Problems. In Soft Computing vol 13 no. 3 pp 307-318.