UNIVERSIDAD DE CIENFUEGOS

“CARLOS RAFAEL RODRÍGUEZ”

FACULTAD DE INGENIERÍA

CARRERA INGENIERÍA INFORMÁTICA

Paquetes informáticos para la herramienta Weka.

Autor: Yinier Rojas Martínez

Tutores: Lic. Ciro Rodríguez León

Ing. Luis Enrique García Hernández

Consultante: Dr. Eduardo Concepción Morales

Cienfuegos, Curso 2014-2015

Agradecimientos

A todas aquellas personas que me han dado su apoyo incondicional y que de una forma u otra han aportado su granito de arena para hacer realidad este trabajo y sobre todas las cosas por ayudarme a ser la persona que soy.

A los que me dejaron soñar y a los que me dieron fuerzas para hacer realidad esos sueños.

A mis tutores y amigos Ciro y Luis Enrique.

A ese grande, Eduardo Concepción.

A mis padres y a mi hermana.

A Edy mi amigo, mi hermano.

A todos

Gracias.

Dedicatoria

A mi familia

A mis amigos

Y a todos aquellos que no tienen miedo de soñar

Resumen

En las últimas décadas el volumen de información ha crecido de forma acelerada, por lo que el análisis manual de los datos se ha hecho cada vez más difícil, impulsando el surgimiento de la Minería de Datos. A la par del surgimiento de esta rama de la Inteligencia Artificial han sido creadas numerosas herramientas para el apoyo en su trabajo.

La presente investigación se desarrolla en torno a la minería de datos, específicamente en la implementación de funcionalidades que no están actualmente disponibles en la herramienta Weka y que sirven como complemento para el trabajo: “Procedimiento para la edición de conjuntos de entrenamiento en problemas no supervisados”.

Weka es una plataforma de software libre para el aprendizaje automático y minería de datos, escrito en Java y desarrollado en la Universidad de Waikato. Soporta varias tareas estándar, especialmente, pre-procesamiento de datos, agrupamiento, clasificación, regresión, visualización, y selección. Para la mayoría de los usuarios las funcionalidades que brinda la herramienta son suficientes en cambio los investigadores podrían querer agregar nuevas funcionalidades para tener más argumentos a la hora de realizar una comparación entre los resultados obtenidos para un experimento dado.

Con las características de Weka para el descubrimiento automático de nuevas clases se facilita el proceso de agregar nuevas funcionalidades a la misma. Este trabajo va a permitir enriquecer la herramienta con nuevas funcionalidades para el trabajo de minería de datos con conjuntos de entrenamientos no supervisados.

Palabras claves: Weka, Inteligencia Artificial, Minería de datos.

Abstract

In recent decades the volume of information has grown rapidly, so manual data analysis has become increasingly difficult, encouraging the emergence of data mining. Along with the emergence of this branch of Artificial Intelligence they have been created numerous tools to support their work.

This research was developed around the mining of data, specifically in implementing features not currently available in the Weka tool and serve as a complement to the work: "Procedure for the issue of joint training problems unsupervised ".

Weka is a free software platform for machine learning and data mining, written in Java and developed at the University of Waikato. It supports several standard tasks, especially, data preprocessing, clustering, classification, regression, visualization and selection. For most users the functionality provided by the tool are sufficient however researchers may want to add new functionality to have more arguments when making a comparison between the results for a given experiment.

With the features of Weka for automatic discovery of new classes the process of adding new features to it is provided. This work will allow the tool to enrich with new features for working with data mining unsupervised training sets.

Keywords: Weka, Artificial Intelligence, Data Mining**.**

Tabla de contenido

Introducción

La Inteligencia Artificial está presente cada día más en la vida de los seres humanos, debido a que los mismos han visto la posibilidad de dotar las máquinas de cierta “inteligencia” y aprovechar de manera ventajosa su uso en las diversas actividades del quehacer cotidiano.

La Inteligencia Artificial (IA) que dio los primeros pasos en los años 50, es una subdivisión de las ciencias de la computación dedicada a crear software y hardware para computadoras que imitan la mente humana. Su principal objetivo es hacer las computadoras más inteligentes, creando software que permitan a una computadora imitar algunas de las funciones del cerebro en áreas de una aplicación seleccionada. La idea no es reemplazar a los seres humanos sino proveerlos de una poderosa herramienta para asistirlos en su trabajo.

La incapacidad del hombre para procesar y extraer nueva información de grandes cantidades de datos sin la ayuda de alguna herramienta da al traste con el surgimiento de una importante aplicación de la Inteligencia Artificial: la Minería de Datos. La Minería de Datos (MD) o KDD (KnowledgeDiscovery in Databases) como se le comenzó a llamar a inicios del año 1996, es definida como el proceso de extraer conocimiento útil y comprensible, previamente desconocido, desde grandes cantidades de datos almacenados en distintos formatos [1].

La minería de datos permite que los datos pasen de ser un "producto" a ser una "materia prima" que hay que explotar para obtener el verdadero "producto elaborado", el conocimiento [2]. Dado que la minería de datos excede la capacidad humana para el análisis de grandes volúmenes de datos, la utilización plena de los datos almacenados depende del uso de técnicas del descubrimiento del conocimiento [3], entre las que se encuentran las de IA.

A la par del desarrollo de la minería de datos han surgido numerosas herramientas para facilitar su trabajo. Actualmente existen varias que por sus características suelen ser las más utilizadas por la comunidad científica, entre ellas, Orange[1] para minería de base de datos y aprendizaje automático basado en componentes para pre-procesamiento de datos, característica de puntuación y filtrado, modelado, evaluación del modelo, y técnicas de exploración. RapidMiner[2] es un ambiente de experimentos en aprendizaje automático y minería de datos que permite a los experimentos componerse de un gran número de operadores anidables arbitrariamente, que se detallan en archivos XML. JHepWork[3] diseñado especialmente para ploteos científicos interactivos en 2D y 3D y contiene bibliotecas científicas numéricas implementadas en Java para

[1] http://www.ailab.si/orange
[2] http://rapidminer.com/
[3] http://jwork.org/jhepwork/

funciones matemáticas, números aleatorios, y otros algoritmos de minería de datos. KNIME[4] para integración de datos, procesamiento, análisis, y exploración además ofrece a los usuarios la capacidad de crear de forma visual flujos o tuberías de datos, ejecutar selectivamente algunos o todos los pasos de análisis, y luego estudiar los resultados, modelos y vistas interactivas. Weka[5] una plataforma para el aprendizaje automático y minería de datos que soporta varias tareas típicas, especialmente pre-procesamiento de datos, agrupamiento, clasificación, regresión, visualización y características de selección.

Debido a que las metodologías y procedimientos de MD de la literatura son todas de propósito general, no conteniendo en ellas especificaciones como las técnicas y algoritmos a utilizar en cada problema específico, así como la importancia del proceso de agrupamiento de instancias sin previa clasificación se crea el trabajo **"Procedimiento de Minería de Datos para problemas no supervisados"**.

No importa cuán completa sea una herramienta, es casi seguro que en un proceso de MD necesitemos algo que no tenemos en ella. Es por eso que muchas veces el camino más corto a la solución de ese problema es añadirle la funcionalidad necesaria a la aplicación.

Entre las herramientas estudiadas y que por sus características permiten ser fácilmente extensibles ninguna agrupa los pasos descritos en el procedimiento antes mencionado. Weka es una herramienta a cual se le pueden añadir nuevas funcionales fácilmente, además de que es Software libre, altamente portable por estar desarrollada en java y una de las que más métodos de MD agrupa; por lo que Selecciona para incorporarle funcionalidades para que sirva de apoyo en la edición de conjuntos de entrenamiento en problemas no supervisados.

Es por ello que teniendo en cuenta todo lo planteado anteriormente se define como:

Problema científico: Ninguna de las herramientas estudiadas agrupan las funcionalidades necesarias para realizar cada uno de los pasos del procedimiento propuesto en el trabajo: "Procedimiento de Minería de Datos para problemas no supervisados".

Objeto de estudio: Algoritmos implementados en la herramienta Weka.

Campo de acción: Implementación de funcionalidades para la herramienta Weka.

Idea a defender: Añadiéndole nuevas funcionalidades específicas a la herramienta Weka se convertirá en el software idóneo para implementar el procedimiento descrito en "Procedimiento de Minería de Datos para problemas no supervisados".

[4] http://www.knime.org/
[5] http://www.cs.waikato.ac.nz/ml/weka/

Objetivo general: Desarrollar un grupo de paquetes con algoritmos de minería de datos que contribuyan a que la herramienta Weka sea más completa para realizar el procedimiento propuesto en el trabajo “Procedimiento de Minería de Datos para problemas no supervisados”.

Objetivos específicos:

- Analizar le estructura de clases de la herramienta Weka
- Seleccionar los métodos a implementar en Weka
- Implementar los métodos seleccionados en Weka
- Validar los métodos incorporados en Weka

Tareas a desarrollar:

- Realizar una revisión bibliográfica de temas afines a la minería de datos y la herramienta Weka
- Realizar un estudio de la metodología de desarrollo para añadir nuevas funcionalidades a la herramienta Weka
- Realizar una revisión bibliográfica de las funciones y métodos seleccionados para implementar
- Implementar las funciones de distancia seleccionadas
- Implementar los métodos de validación de agrupamiento seleccionados
- Realizar experimentos con diferentes fuentes de datos para probar el correcto funcionamiento de los paquetes incorporados a Weka

Aporte práctico:

El presente trabajo va a permitir enriquecer la herramienta Weka con nuevas funcionalidades que permitan una mayor diversidad de opciones a la hora de generar clústeres así como validar los mismos. Estas nuevas funcionalidades permitirán su uso para realizar el procedimiento propuesto en el trabajo “Procedimiento de Minería de Datos para problemas no supervisados”.

Métodos de Investigación

Teóricos

Método Histórico Lógico: Se utiliza con el objetivo de profundizar en los antecedentes de las teorías correspondientes a la minería de datos, en su de cursar histórico, revelando las características y tendencias fundamentales.

Método Analítico–Sintético: Se utilizará para captar y resumir varios documentos en los cuales se describen procedimientos utilizados durante el proceso de minería de datos. De ellos se extraerán las ideas fundamentales para su posterior uso en la solución del problema.

Resumen capitular

Para la estructura de este trabajo se proponen 3 capítulos. Todos ellos constan de un epígrafe introductorio para comentar de forma general lo que se estará tratando a lo largo del capítulo y unas conclusiones parciales del mismo.

El primer capítulo, nombrado ***"Fundamentos Teóricos"*** tiene como objetivo plasmar todos los conocimientos necesarios para llevar a cabo este trabajo. Por lo cual se estará hablando de forma general de la Minería de datos y profundizando aún más en las funciones de distancia y los índices de validación. También se hablará sobre los paquetes propuestos para incorporar a Weka, así como una descripción del funcionamiento de estos. Por último se explican las características y ventajas de Weka para incorporar nuevos paquetes.

El segundo capítulo lleva por nombre ***"Descripción de la solución propuesta"*** Está diseñado para presentar la solución dada al problema definido para este trabajo. Se muestra de forma detallada como quedaron los paquetes para incorporar a Weka. Además se describen las acciones necesarias para poder ampliar el uso de la herramienta Weka con la incorporación de los nuevos paquetes.

El capítulo 3, ***"Resultados Experimentales"***, va a plasmar los resultados que fueron obtenidos luego de incorporar a Weka los nuevos paquetes. Se analizarán los resultados que se obtuvieron sobre los datos.

Capítulo 1 “Fundamentos Teóricos”

En este capítulo se detallarán los aspectos teóricos necesarios para llevar a cabo este trabajo. Se hablará de forma general de la Minería de Datos y los métodos de agrupamiento, profundizándose más en las Funciones de Distancia que utilizan dichos métodos y en los Índices de Validación. Se explicará por qué incorporar a Weka los paquetes propuestos, detallando los mismos. Para finalizar se describen las características y aspectos más relevantes de la herramienta.

Minería de datos

La minería de datos es un campo de las ciencias de la computación referido al proceso que intenta descubrir patrones en grandes volúmenes de datos. Utiliza los métodos de inteligencia artificial, con el objetivo extraer información de un conjunto de datos y transformarla en una estructura comprensible para su uso posterior.

Un proceso típico de minería de datos consta de los siguientes pasos generales:

- Selección del conjunto de datos
- Análisis de las propiedades de los datos
- Transformación del conjunto de datos de entrada
- Seleccionar y aplicar la técnica de minería de datos
- Extracción de conocimiento
- Interpretación y evaluación de datos

Métodos de Agrupamiento

En esta sección abordaremos los algoritmos de agrupamiento más conocidos y utilizados en la comunidad científica. Para una mejor comprensión los dividiremos y analizaremos en 3 grupos: particionales, jerárquicos y redes neuronales artificiales, puesto que hay un gran número de estas últimas que se utilizan para tareas de agrupamiento.

Particionales

Los métodos de agrupamiento particionales son llamados así puesto que al culminar su ejecución cada objeto del conjunto de datos pertenece a uno y solo a uno grupo de los grupos conformados. Es decir si denotamos $O = \{O_{1,} O_{2,} \dots, O_{n,}\}$ al conjunto de n objetos, se trata de dividir en k grupos$Cl_{1,}, \dots, Cl_k$de tal forma que:

- $\bigcup_{j=1}^{k} Cl_j = O$
- $Cl_j \cap Cl_j = \emptyset \; \forall i \neq j$

Tratando de que los objetos que pertenecen a un mismo cluster sean semejantes entre sí y a la vez disjuntos a los restantes [4].

Jerárquicos

Los métodos jerárquicos crean una descomposición jerárquica de los objetos, conjunto de datos de entrada formando un dendograma. Esto no es más que un árbol que divide la base de conocimiento en muestras más pequeñas recursivamente. Este gráfico puede ser formado de dos formas de abajo hacia arriba o desde arriba hacia abajo. Estos algoritmos combinan o dividen los grupos existentes creando una estructura escalonada que refleja el orden en el que los grupos van siendo tratados.

El esquema de abajo hacia arriba (*bottom up*), también llamado método aglomerativo, comienza con cada objeto siendo parte de un grupo independiente. Luego va mezclando secuencialmente los objetos o grupos de acuerdo con alguna medida, como distancia entre dos centros de dos grupos, hasta que todos los elementos estén en un solo grupo o hasta que se cumpa una condición de parada [5].

El otro esquema (*top down*), también llamado divisor, comienza con todos los objetos en un cluster. En cada iteración sucesiva los grupos son divididos en clusters más pequeños de, acuerdo a alguna medida, hasta que al final cada objeto es un cluster o se llega a una condición de parada [5].

Redes neuronales artificiales

Una Red Neuronal Artificial (RNA), además de los pesos y las conexiones, cada neurona tiene asociada una función matemática denominada función de transferencia. Dicha función genera la señal de salida de la neurona a partir de las señales de entrada. La entrada de la función es la suma de todas las señales de entrada por el peso asociado a la conexión de entrada de la señal. Algunos ejemplos de entradas son la función escalón, la lineal, la sigmoide y la función gaussiana, recordando que la función de transferencia es la relación entre la señal de salida y la entrada [6].

Funciones de distancia y similitud.

Antes de realizar el agrupamiento es necesario determinar una medida o función de distancia o de similitud a usar. La función refleja el grado de cercanía o separación de los objetos y deben corresponderse con las características que se creen distinguen a los grupos contenidos en los datos. En muchos casos esta característica es dependiente de los datos o el contexto del problema en cuestión, y no existe una función que universalmente sea la mejor para todo tipo de problema de agrupamiento. Por lo que es crucial, para el buen desempeño de una técnica de este tipo, la elección de una correcta función de distancia [7][8].

Toda función de distancia debe cumplir las siguientes condiciones [4][8][9]:

- La distancia debe ser simétrica, es decir, la distancia de un objeto x a otro y es la misma que la de y al x. Esto es: $d(x, y) = d(y, x)$
- La distancia entre cualesquiera dos objetos tiene que ser positiva (positividad), esto es: $d(x, y) \geq 0$

Si además cumple las siguientes condiciones entonces la función es considerada una métrica:

- La función debe satisfacer la desigualdad triangular, que es: $d(x, z) \leq d(x, y) + d(y, z)$
- La distancia entre dos objetos debe ser cero si y solo si los dos objetos son idénticos (reflexividad), esto es: $d(x, y) = 0 \Leftrightarrow x = y$

Del mismo modo una función de similitud es aquella que cumple las siguientes condiciones [9][4]:

- Simetría: $d(x, y) = d(y, x)$
- Positividad: $0 \leq d(x, y) \leq 1, \forall\, x, y$

Veremos ahora un acercamiento a varias funciones de distancia y similitud, sus definiciones matemáticas, su semántica y los tipos de algoritmos y datos en los que se desempeñan mejor.

Coseno

Es una medida de similitud, no es considerada una métrica puesto que no cumple con la desigualdad triangular. Evalúa la similitud entre dos vectores con el valor del coseno del ángulo comprendido entre ellos. Es la más común utilizada en minería de textos como un indicador de cohesión de los grupos, puesto que es independiente del tamaño de dicho vector. Su definición es:

$$S(i, j) = \frac{X_i^T X_J}{\left||X_i|\right| \left||X_j|\right|}$$

Es decir, el producto vectorial entre los dos vectores dividido entre la multiplicación de sus normas euclidianas.

Esta función trigonométrica proporciona un valor igual a 1 si el ángulo comprendido es cero, es decir si ambos vectores apuntan a un mismo lugar. Cualquier ángulo existente entre los vectores, el coseno arrojaría un valor inferior a uno. Si los vectores fuesen ortogonales el coseno se anularía, y si apuntasen en sentido contrario su valor sería -1. De esta forma, el valor de esta medida se encuentra entre [-1,1] [10][11].

Coeficiente de correlación de Pearson

El coeficiente de correlación de Pearson es una medida de la relación lineal entre dos variables aleatorias cuantitativas. Esta es independiente de la escala de medida de las variables. Podemos definir el coeficiente de correlación de Pearson como un índice que puede utilizarse para medir el grado de relación de dos variables siempre y cuando ambas sean cuantitativas [12]. Su cálculo sería:

$$r_{ij} = \frac{\sum_{i=1}^{n} x_i y_{i-} \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{\sqrt{\sum_{i=1}^{n} {x_i}^2 - (\sum_{i=1}^{n} x_i)^2}\sqrt{\sum_{i=1}^{n} {y_i}^2 - (\sum_{i=1}^{n} y_i)^2}}$$

El valor del índice de correlación varía en el intervalo [-1,1]:

Si r = 1, existe una correlación lineal positiva perfecta.

Si 0 < r < 1, existe una correlación lineal positiva.

Si r = 0, no existe relación lineal.

Si -1 < r < 0, existe una correlación lineal negativa.

Si r = -1, existe una correlación lineal negativa perfecta.

Pero cuando este coeficiente es usado como medida de distancia lo importante no es el signo sino el valor modular por eso podemos modificar la fórmula de su cálculo a [4]:

$$D(x_i, x_j) = (1 - r_{ij})/2$$

En [13] se dice que esta medida es muy apropiada para usar con los mapas auto-organizados.

Medidas de validación del agrupamiento

Son muchas las técnicas desarrolladas para la validación de los conglomerados. Estas estrategias son denominadas índices y se pueden clasificar en externos o internos. Los que utilizan una partición de referencia obtenida de manera independiente son los primeros, y en internos, los que utilizan información que se obtiene a partir del mismo proceso de clasificación [14].

Dado que en la problemática que define la presente investigación nunca se tiene presente una clasificación de referencia de los datos, nos centraremos en estudiar los índices de validación internos. Estos permiten verificar si la estructura del cluster producido por un algoritmo de agrupamiento coloca adecuadamente los datos, pero usando solamente información inherente a la base de datos [14].

Dunn

Dunn [15] corresponde al radio de la distancia más pequeña entre las observaciones de diferentes clusters y la distancia inter-cluster más grande. Tiene valores entre 0 e infinito y debe maximizarse para que la agrupación sea la óptima. Semánticamente puede decirse que mide cuan compactos y separados están los clusters entre ellos. Es definido como:

$$D = \min_{i=1,\ldots,m} \left\{ \min_{j=i+1,\ldots,m} \left(\frac{d(C_i, C_j)}{\max_{k=1,\ldots,m} diam(C_k)} \right) \right\}$$

Donde la función de diferencia entre dos clusters Ci y Cj es $d(C_i, C_j) = \min_{x \in C_i, y \in C_j} d(x, y)$ y el diámetro de un cluster C es definido como $diam(C) = \max_{x,y \in C} d(x, y)$

Calinski y Harabasz

Creado por [16] está basado en la suma cuadrática interna y entre clusters. Utiliza las matrices de dispersión y el valor que maximice esta función será el candidato para especificar el número de clusters que se usará para clasificar los datos. Si se tiene una partición con k clusters el índice de Calinski y Harabasz se calcula de la siguiente manera:

$$CH(k) = \frac{tr(S_B)/(k-1)}{tr(S_W)/(n-k)}$$

S_B y S_W son las matrices de dispersión externa e interna respectivamente:

$$S_W = \sum_{k=1}^{k} \sum_{i \in P_k} (x_i - \bar{x}_k)^T (x_i - \bar{x}_k)$$

$$S_B = \sum_{k=1}^{k} n_k (x_i - \bar{x}_k)^T (x_i - \bar{x}_k)$$

Ball and Hall

Introducido por [17], se basa en las matrices de dispersión interna y externa. Su fórmula de cálculo es: SSW/k donde k es el número de clusters y SSW es la suma de cuadrados dentro de los clusters. El máximo valor de las segundas diferencias respecto al de la izquierda es tomado como el número de clusters apropiado [18].

Hartigan

Este índice se basa en las matrices de dispersión interna (SSW) y externa (SSB). Su fórmula de cálculo es:$\log(SSB/SSW)$El máximo valor será considerado como el número de clusters apropiado [18]

Davies-Bouldin

Este índice de validación trata de maximizar la distancia entre clusters a partir de minimizar la distancia entre los elementos de cada cluster y su centroide. Su fórmula de cálculo es: $1/k \sum_{i=1}^{k} R_i$ donde $R_i$es el máximo valor de R_{ij} para $i \neq j$ y su fórmula es $(SSW_i + SSW_j)/DC_{ij}$ y DC_{ij} es la distancia entre el centroide$i$ y el centroide$j$. El mínimo valor es tomado como el número de clusters apropiado [18].

Herramienta Weka

Escrito en Java, Weka (Entorno Waikato para el Análisis del Conocimiento) es una conocida suite de software para máquinas de aprendizaje que soporta varias tareas típicas de minería de datos, especialmente pre procesamiento de datos, agrupamiento, clasificación, regresión, visualización y características de selección. Sus técnicas se basan en la hipótesis de que los datos están disponibles en un único archivo plano o relación, donde cada punto marcado es etiquetado por un número fijo de atributos. WEKA proporciona acceso a bases de datos SQL utilizando conectividad de bases de datos Java y puede procesar el resultado devuelto como una consulta de base de datos. Su interfaz de usuario principal es el Explorer, pero la misma funcionalidad puede ser accedida desde la línea de comandos o a través de la interfaz de flujo de conocimientos basada en componentes [19]. Una de las suites más utilizadas en el área en los últimos años, destaca su facilidad para añadir extensiones y modificar métodos gracias a su filosofía de paquetería para lograr esto sin la necesidad de modificar el núcleo de la aplicación. Es multiplataforma ya que al estar completamente implementada en java puede correr en casi cualquier plataforma, solo es necesario tener la máquina virtual de java. Es Software Libre, distribuida bajo la licencia GNU-GPL, la más ampliamente usada en el mundo del software, garantiza a los usuarios finales la libertad de usar, estudiar, compartir y modificar el software. Estas razones hacen que la herramienta sea la seleccionada para llevar a cabo el presente trabajo.

Conclusiones Parciales

En este capítulos hemos hecho un estudio del arte acerca de la minería de datos particularizando en las áreas que competen a nuestra investigación: las funciones de distancia y los índices de validación. También hicimos un análisis de la herramienta Weka puesto que el objetivo es incorporarle funcionalidades del procedimiento mencionado.

Capítulo 2 “Descripción de la solución propuesta”

En este capítulo se describe la forma de extender la herramienta Weka, detallando la estructura de cada uno de los paquetes a incorporarle, así como la implementación de las nuevas funcionalidades teniendo en cuenta los requisitos funcionales.

¿Cómo extender Weka?

Gracias al descubrimiento automático de clases de la herramienta es muy fácil añadirle nuevas funcionalidades, a continuación se describen los aspectos a tener en cuenta a la hora de hacerlo.

Estructura de un paquete.

Un paquete para Weka es un archivo compactado (zip) que debe ser extraído en el directorio que crea por defecto la herramienta al instalarse: *C:\Users\UserName\wekafiles\packages*, su estructura puede apreciarse en la Figura 1.

```
<current directory>
  +-DTNB.jar
  +-Description.props
  +-build_package.xml
  +-src
  |   +-main
  |   |   +-java
  |   |       +-weka
  |   |           +-classifiers
  |   |               +-rules
  |   |                   +-DTNB.java
  |   +-test
  |       +-java
  |           +-weka
  |               +-classifiers
  |                   +-rules
  |                       +-DTNBTest.java
  +-lib
  +-doc
```

Figura 1. Estructura de un paquete.

El fichero *Description.props* contiene la información básica referente al paquete, como el nombre, el autor, una breve descripción, las dependencias, etc. Algunos campos son obligatorios y otros opcionales. El fichero *build_package.xml* también contiene información del paquete, en el sitio oficial de Weka se puede encontrar plantillas de dichos ficheros.

En el directorio *doc* se debe poner la documentación necesaria para la comprensión de la nueva funcionalidad. Es importante también que el paquete tenga al menos un compilado *jar* en la raíz y si es necesario para el funcionamiento del mismo alguna otra biblioteca, se ubicará en el directorio *lib* como un compilado igualmente. Al ejecutarse Weka esta cargará todos estos ficheros de forma automática, conteniendo las nuevas funcionalidades extendidas.

Conceptos fundamentales

ARFF

Un archivo ARFF (*Attribute-Relation File Format*) es un archivo de texto ASCII muy similar a un fichero *CSV*(separado por comas) que describe una lista de instancias (*Instances*) compartiendo un conjunto de atributos y que contiene cierta información adicional en la cabecera del mismo.

Instances

Es una estructura de datos en forma de tabla en la que cada fila representa una instancia (*Instance*) y cada columna un conjunto de atributos (*Attribute*).

Instance

Representa una fila de las instancias que contiene un conjunto de atributos.

Attribute

Contiene la información única referente a un elemento específico.

Para una mejor comprensión de la estructura de un archivo *arff* ver el ejemplo en el Anexo 1**.**

Requisitos funcionales

- Implementar función de distancia Coseno
- Implementar función de distancia Coeficiente de Correlación de Pearson
- Implementar índice de validación Dunn
- Implementar índice de validación Ball and Hall
- Implementar índice de validación Hartigan
- Implementar índice de validación Davies Bouldin
- Implementar índice de validación Calinski and Harabasz
- Implementar la extensión de la Interfaz Gráfica de Usuario (GUI)

Implementar función de distancia Coseno

Para implementar una nueva función de distancia y que esta sea visible para Weka es necesario que la nueva clase definida para dicha función herede de la clase *weka.core.NormalizableDistance,* de esta manera la nueva función poseerá las características y métodos necesarios que harán posible su usabilidad por la herramienta.

En la Figura 2 se muestra el diagrama de clases para el requisito funcional y posteriormente se describen las funciones más importantes de las clases.

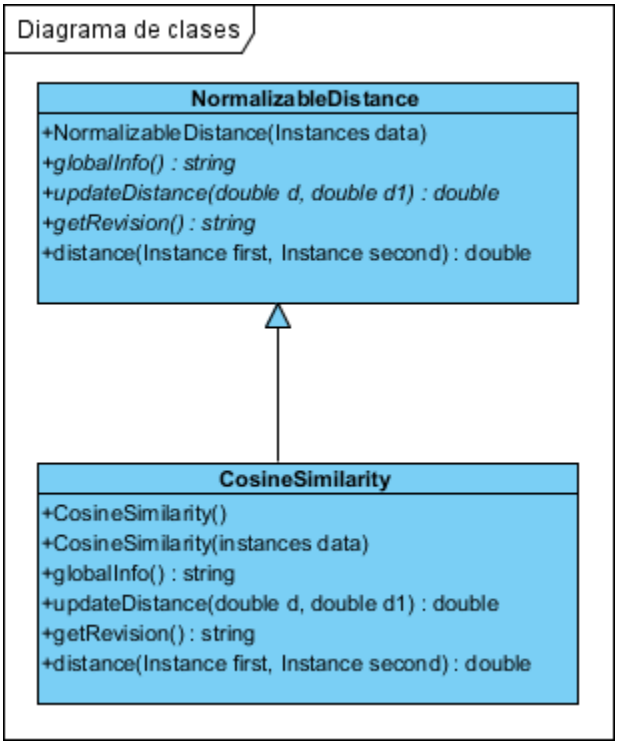


Figura 2 Diagrama de clases *CosineSimilarity*

globalInfo(): Este método abstracto de la clase *NormalizableDistance* debe ser implementado en la clase declarada para la función de distancia y devuelve una cadena (*String*) con información referente a dicha clase para ser mostrado en la interfaz gráfica de usuario. La longitud de la descripción es a consideración personal, pero debe ser suficiente para entender el nuevo algoritmo en cuestión.

getRevision(): Este método abstracto de la clase *NormalizableDistance* debe ser implementado en la clase declarada para la función de distancia y devuelve una cadena (*String*) con información referente a la revisión del código fuente de la clase.

updateDistance(): Este método abstracto de la clase *NormalizableDistance* debe ser implementado en la clase declarada para la función de distancia y recibe dos valores

(*double*) para calcular la distancia entre ellos cada vez que sea llamado dentro de la función de distancia (*distance*).

distance(Instance first, Instance second): Este método reescrito en la clase declarada para la función de distancia recibe como parámetros dos valores de tipo *Instance*, cada vez que es llamada por el método de agrupamiento y calcula la distancia entre cada uno de sus elementos de tipo *Attribute*.

Implementar función de distancia Coeficiente de Correlación de Pearson

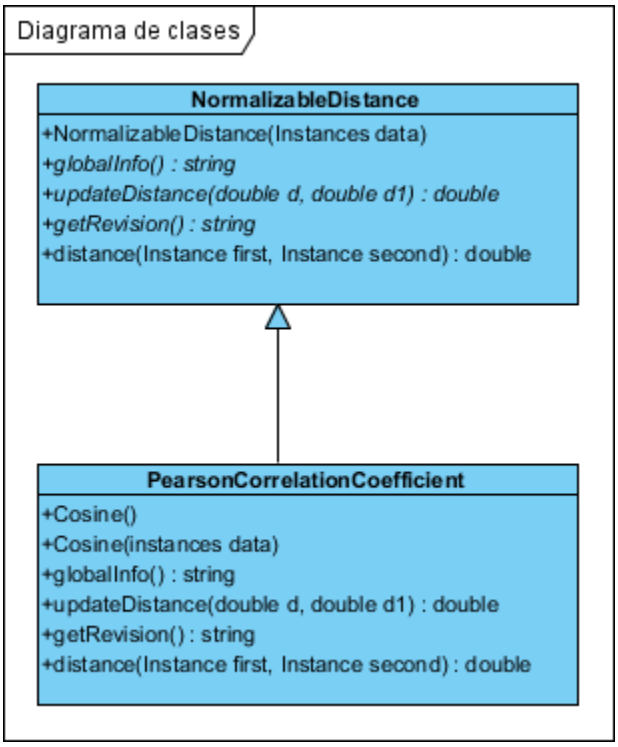


Figura 3. Diagrama de clases *PearsonCorrelatioCoefficient*

La clase *PearsonCorrelationCoefficient* Figura 3, implementa los métodos de la clase *NormalizableDistance* de Weka que se explicaron en la sección anterior.

Una vez que los paquetes han sido creados y ubicados en el directorio específico, al ejecutar nuevamente la aplicación estarán disponibles las nuevas funciones de distancia, como se muestra en Figura 4.

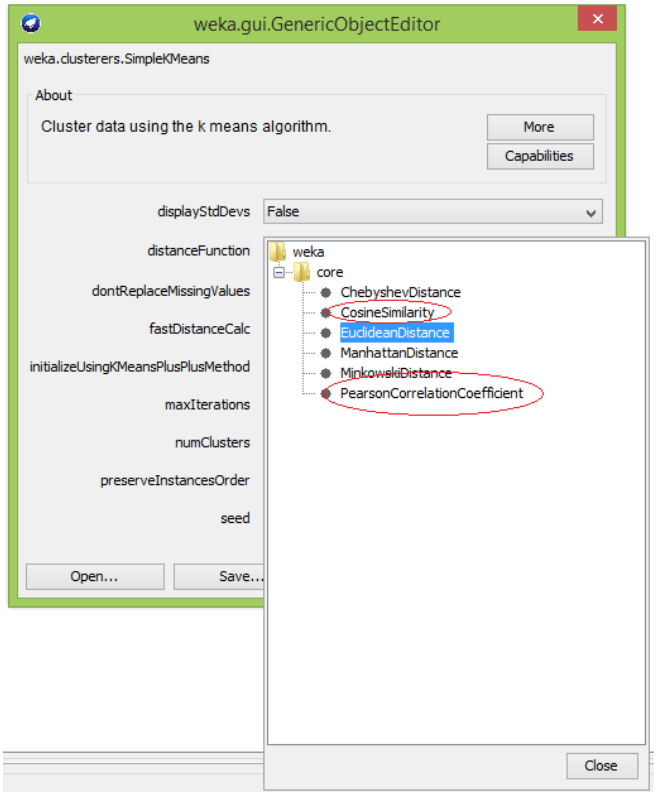


Figura 4. Funciones de distancia añadidas

Implementar índices de validación

Para implementar los requisitos funcionales relacionados con los índices de validación se creó la clase *Validation,* como se muestra en Figura 5en la cual se agrupan los índices, cada uno como una función. En esta clase se definen como variables globales los datos que se utilizan por varios de los índices de validación, dichos datos son calculados al llamar el constructor de la clase, este recibe como parámetro instancias (*Instances*). Es válido aclarar que este tipo de funcionalidad no existía anteriormente en Weka, por lo que no hay clases de las que heredar ni interfaces que implementar. Los principales métodos son:

distance(Instance x, Instance y): calcula la distancia entre los elementos.

min(double[] s): calcula el mínimo valor del arreglo "*s".*

max(double[] s): calcula el máximo valor del arreglo "*s".*

getAssignments(Instances data): devuelve un arreglo de enteros con las asignaciones de las instancias *"data"*.

getCentroids(Instances data, int[] assignments): calcula los centroides para "data" a partir de las asignaciones de los mismos.

removeIgnoreCols(Instances inst, String cols): esta función elimina la columna definida por *"cols"* de las instancias *"inst"*.

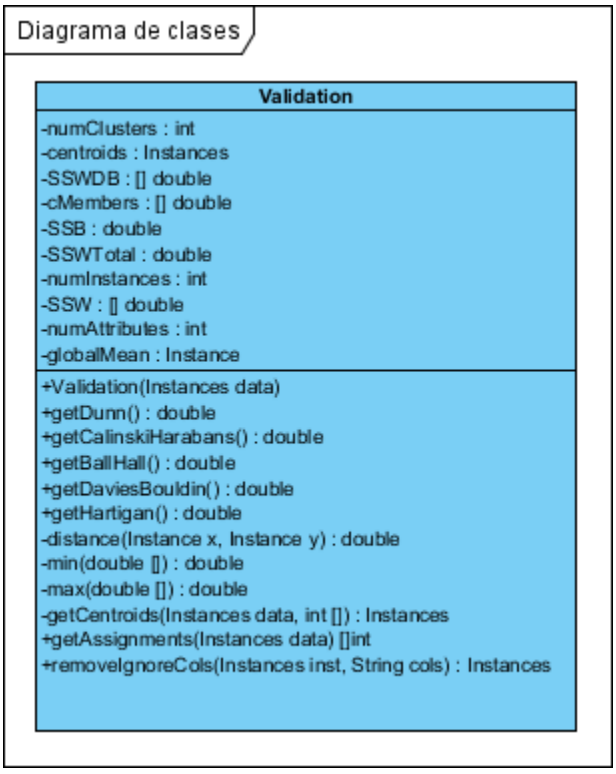


Figura 5. Clase *Validation*

Implementar la extensión de la Interfaz Gráfica de Usuario

Weka no tiene ninguna sección para la validación de los agrupamientos, es por eso que para poder utilizar los índices de validación mencionados anteriormente desde la interfaz gráfica, se le debe añadir un nuevo componente, a partir de las facilidades de extensión que nos brinda la herramienta.

Para añadir un nuevo componente a la interfaz gráfica de Weka es imprescindible que nuestra clase herede de la clase *javax.swing.JPanel* Figura 7 e implemente la interfaz *weka.gui.explorer.Explorer.ExplorerPanel* Figura 8 y además de los ficheros descritos anteriormente que conforman el paquete, dentro de la raíz del mismo debe estar un fichero nombrado *Explorer.props* que se obtiene del directorio de instalación de Weka y que se modifica para añadirle el nombre de nuestro nuevo componente, ver Figura 6. En él mismo lo que aparece precedido del símbolo # es comentario, la línea que añade el nuevo componente es la que está precedida por "*Tabs="* y a continuación la ruta del paquete. Si además se añade al final de la línea "*:standalone"* significa que el componente estará activado sin haber cargado instancias previamente.

```
# Explorer.props file. Adds the ValidationPanel to the Tabs key.

Tabs=weka.gui.explorer.ValidationPanel:standalone,
TabsPolicy=append
```

Figura 6. Ejemplo de fichero *Explorer.props*

A continuación se describen las clases que forman parte de este requisito funcional así como los principales métodos de cada una.

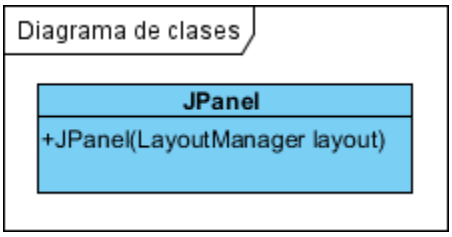


Figura 7. Clase *JPanel*

JPanel(LayoutManager layout): es el constructor que se utiliza de la clase *javax.swing.JPanel,* al cual se le pasa un controlador para ajustes de diseño.

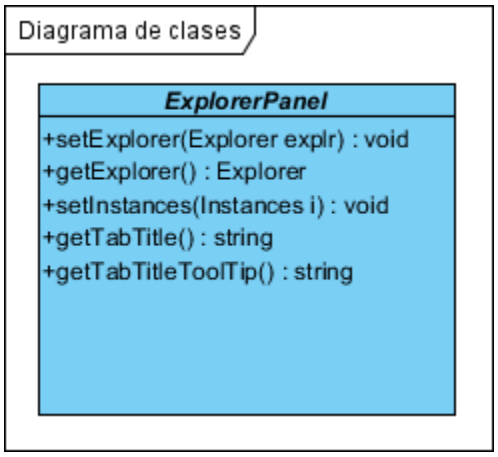


Figura 8. Clase *ExplorerPanel*

setExplorer(Explorer explr): es el método que establece el nuevo componente (*"explr"*) en la interfaz gráfica.

getExplorer(): devuelve el componente que se haya establecido.

setInstaces(Instances i): establece las instancias cargadas en el ámbito global de la aplicación.

getTabTitle(): devuelve el nombre del componente.

getTabTitleToolTip(): devuelve información útil del componente.

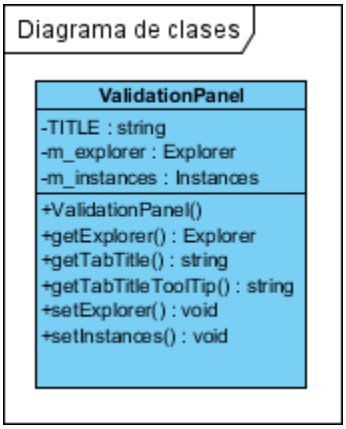


Figura 9. Clase *ValidationPanel*

*ValidationPanel*Figura 9 es la clase definida para el nuevo componente, en ella están presentes los métodos antes descritos.

A continuación se describen otras clases adicionales que forman parte del nuevo componente y de igual manera se describen su funcionalidad y pricipales métodos.

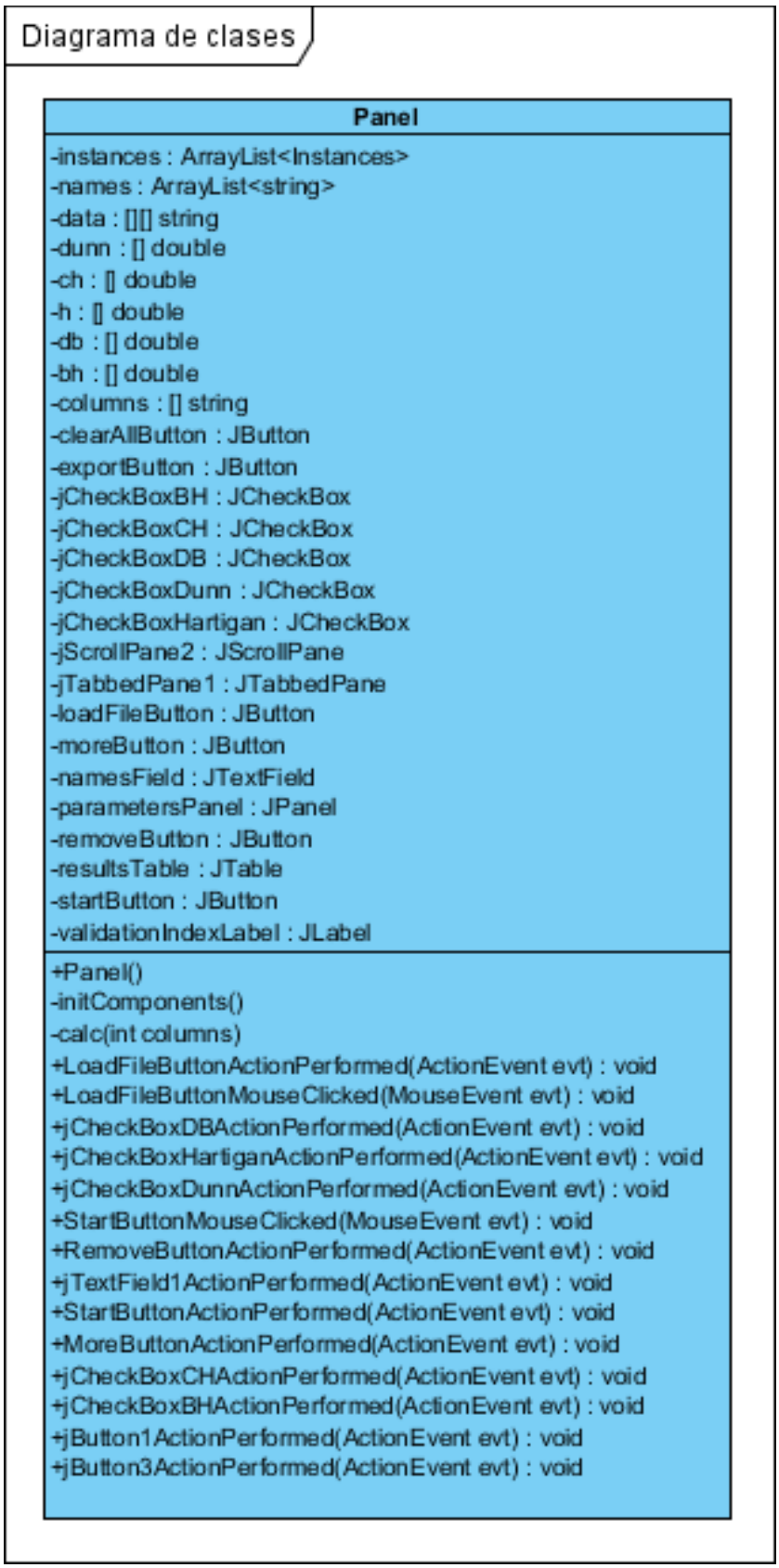


Figura 10. Clase *Panel*

Panel Figura 10, en esta clase se construye el nuevo componente, a cada elemento se le asigna su respectiva función y en la misma se hace uso de las clases *Validation* y *Exporter.*

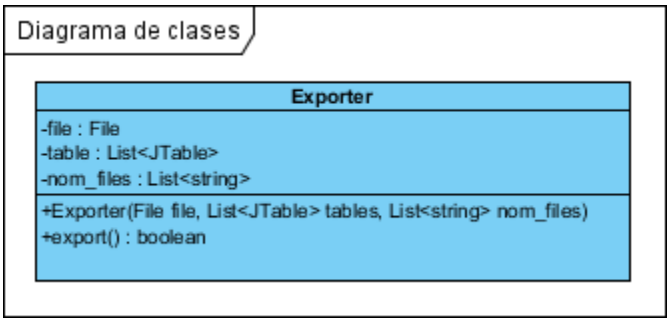


Figura 11. Clase *Exporter*

Exporter Figura 11, es la clase creada para exportar a un fichero *CSV* los resultados obtenidos de la validación, el constructor de la clase recibe una lista con la(s) tabla(s) y el/los nombre(s) de esta(s) en otra lista; tiene un solo método que es el que exporta los datos y devuelve un valor verdadero o falso en dependencia si se pudieron exportar los datos o no.

En la Figura 12se muestra el diagrama de clases de la interfaz gráfica.

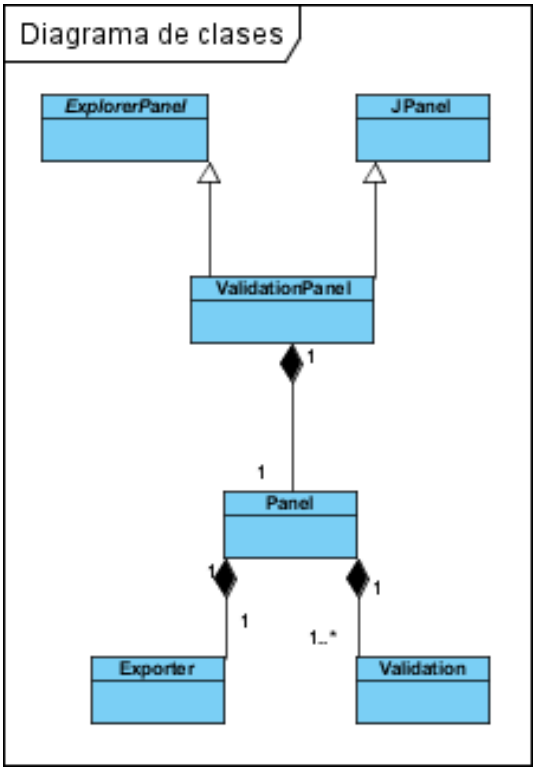


Figura 12.Diagrama de clases de la interfaz gráfica de usuario

Una vez incorporado el paquete *ClusterValidation,* al ejecutar nuevamente la aplicación se puede apreciar la nueva ventana, como se muestra en Figura 13.

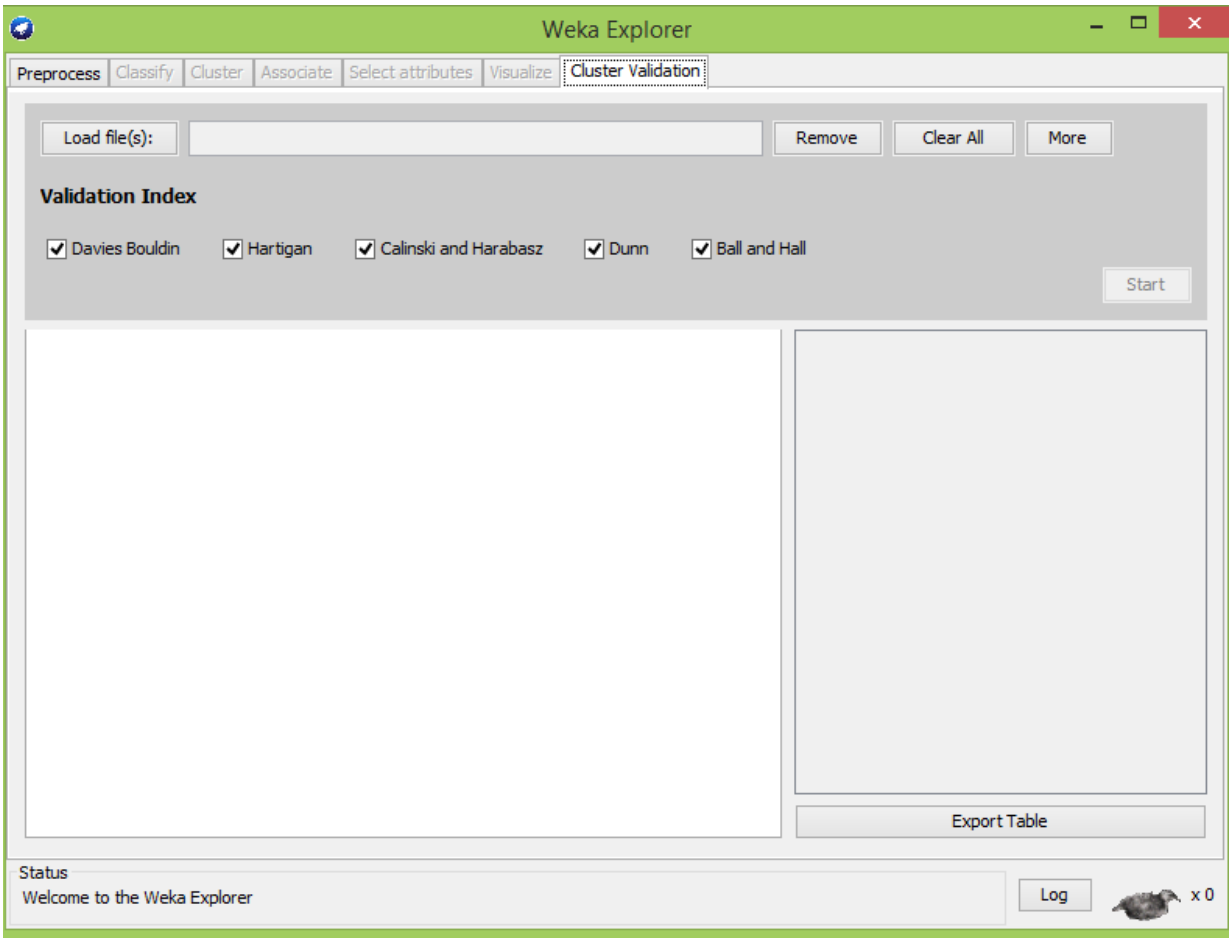


Figura 13. Interfaz gráfica de usuario incorporada a Weka

Conclusiones parciales

En este capítulo se explica la filosofía de extensión de la herramienta Weka, su estructura de paquetes y conceptos fundamentales asociados a ella. Además se detalla acerca de la implementación de los requisitos funcionales.

Capítulo 3 “Resultados Experimentales”

El capítulo 3, “Resultados Experimentales”, va a plasmar los resultados que fueron obtenidos luego de incorporar a Weka los nuevos paquetes. Se analizarán los resultados que se obtuvieron sobre los datos.

Validación de las funciones de distancia

Para validar las nuevas funciones de distancia incorporadas a la herramienta se realizaron dos formas de prueba, la primera consistió en comprobar que calculaban de forma correcta y para esto se utilizaron diferentes juegos de datos que fueron calculados en la herramienta SPSS y luego se calcularon internamente con las funciones obteniéndose resultados similares. Luego se probaron integradas a Weka arrojando resultados satisfactorios.

Validación de los Índices

Para validar los índices añadidos a la herramienta se utilizaron conjuntos de datos correctamente agrupados los cuales se manipularon para agruparlos de forma aleatoria e incorrecta, de esta manera una vez obtenidos los resultados de los índices calculados sobre los datos en cuestión, analizar si los mismos tenían relación con la realidad.

Se sabe que los conjuntos de datos están correctamente agrupados inicialmente pues son bases de casos supervisadas (poseen la clasificación de cada una de sus instancias) ampliamente usadas y de prestigio internacional como se describirán a continuación. Luego se tomaron estos conjuntos de datos y en el atributo clase se pusieron valores aleatorios, por lo que la nueva distribución no será la correcta.

Descripción de los conjuntos de datos.

Iris Plant Database (Iris): Ésta es quizás la mejor base de datos conocida que se pueda encontrar en la literatura de reconocimiento de patrones. Se considera un clásico en este campo y es citada frecuentemente. Los datos contienen 3 clases de 50 instancias cada una, donde cada clase se refiere a un tipo de planta iris. Una clase es linealmente separable de las otras 2 y a su vez esta no es linealmente separable entre sí.

Glass Identification Database (Glass): Esta base de datos fue creada por B. German. Los datos en ella contienen tipos de vidrio. Este estudio fue motivado ya que en una investigación criminal el vidrio encontrado puede usarse como evidencia si este puede ser identificado. Tiene 214 instancias y 7 clases.

Pima Indians Diabetes Database (Diabetes): Esta base de datos es una particularización de un conjunto de datos mayor. En la misma los datos son todos

referentes a personas del sexo femenino con al menos 21 años de edad. Tiene 768 instancias y 2 clases.

Estructura de los experimentos

Cada una de las bases de datos anteriores fueron desagrupadas varias veces, asignando las instancias a clases distintas, como se explica en la Figura 14, creando así varios agrupaciones a partir del original, manteniendo el orden de los datos y modificando la clase a la cual pertenecen los mismos.

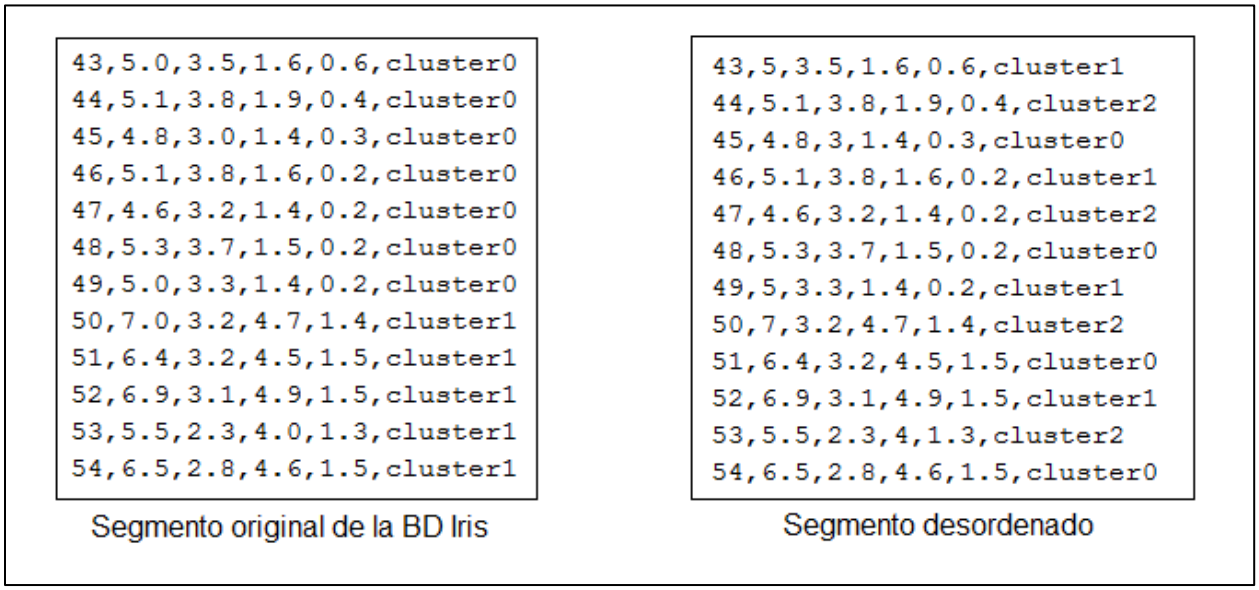


Figura 14. Ejemplo de experimento creado para la validación de los índices

Resultados obtenidos

A continuación se presentan las tablas obtenidas en cada uno de los experimentos y la y una figura de la pantalla con los resultados para cada una de las bases de datos anteriores resaltando el resultado correcto en cada caso. En las mismas aparece en primer orden la BD original y posteriormente aparecen los nombres de las BD desordenadas, que se componen del nombre original seguido de “Rx” donde x representa el número que se escogió para desordenar la misma. En la Figura 14 el nombre del fichero desordenado queda irisR3 ya que solo hay tres clases y por eso aparecen de forma de forma consecutiva.

P/I	Davies Bouldin	Hartigan	Calinski and Harabasz	Dunn	Ball and Hall
iris.arff	**4.284824454**	**0.157146242**	**86.00725456**	**0.00075553**	**3178.7333336**
irisR10.arff	15.06515996	-0.061030701	69.14838518	0.000272153	3178.7333335
irisR3.arff	99.00197985	-0.073805513	68.27064597	4.72681E-05	3178.7333332
irisR5.arff	30.87153579	-0.071464766	68.43063747	0.000152796	3178.7333333

Tabla 1. Resultados para *Iris*

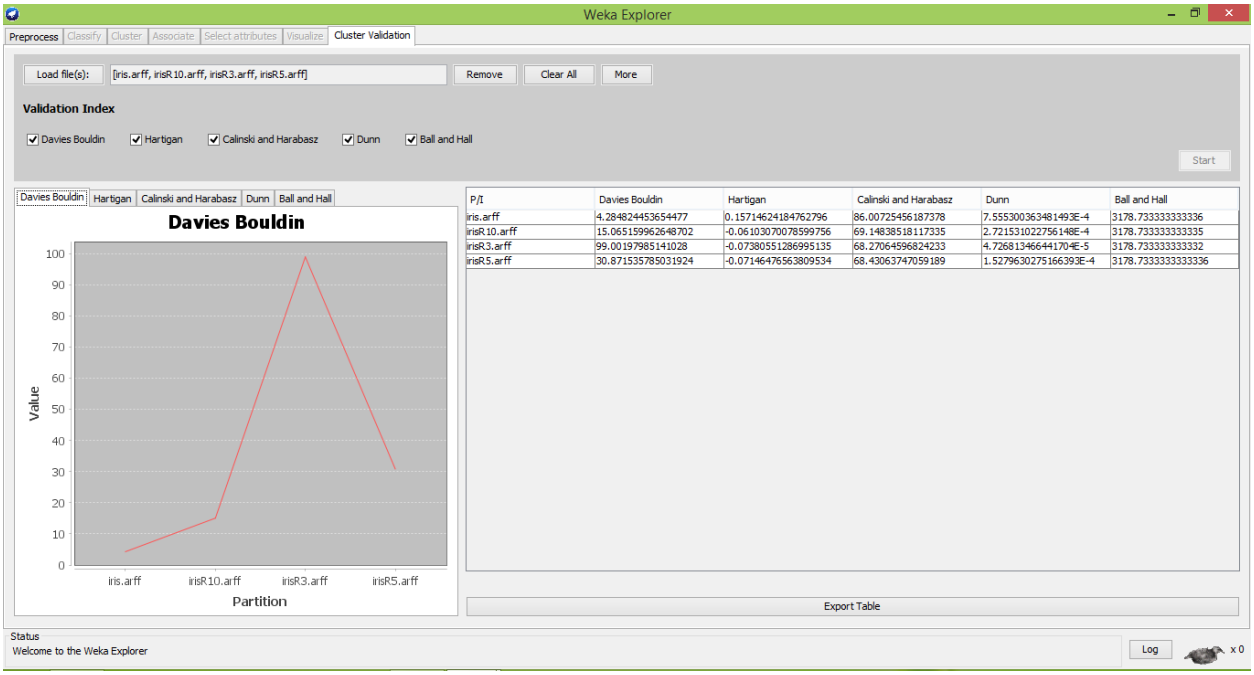


Figura 15. Resultados para *Iris*

P/I	Davies Bouldin	Hartigan	Calinski and Harabasz	Dunn	Ball and Hall
glass.arff	**139.9614801**	**0.000582074**	**37.62189236**	**1.2992E-06**	**179863.9635**
glassR10.arff	219.5274719	-0.000902171	37.56609368	1.90077E-06	179863.9634
glassR2.arff	202.100466	-0.000911838	31.13826068	2.66023E-06	154169.1115
glassR5.arff	239.0386642	-0.00098416	37.56301378	2.42603E-06	179863.9634

Tabla 2. Resultados para *Glass*

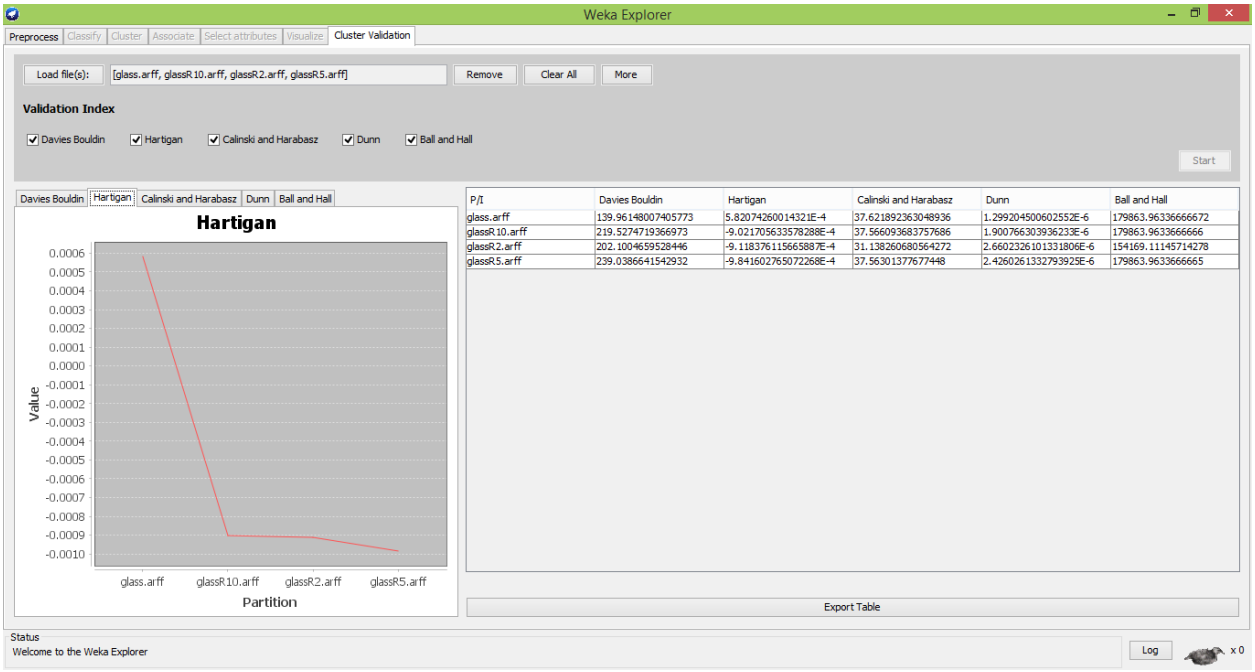


Figura 16. Resultados para *Glass*

P/I	Davies Bouldin	Hartigan	Calinski and Harabasz	Dunn	Ball and Hall
diabetes.arff	**2.80477195**	**-0.259909102**	**496.6023594**	**1.0745E-05**	**13810915.42**
diabetesR10.arff	24.14012172	-0.41567353	424.9728126	1.14257E-06	13810915.42
diabetesR2.arff	13.84164072	-0.410708884	427.0878979	2.05537E-06	13810915.42
diabetesR5.arff	26.18128649	-0.416040266	424.816988	1.07609E-06	13810915.42

Tabla 3. Resultados para *Diabetes*

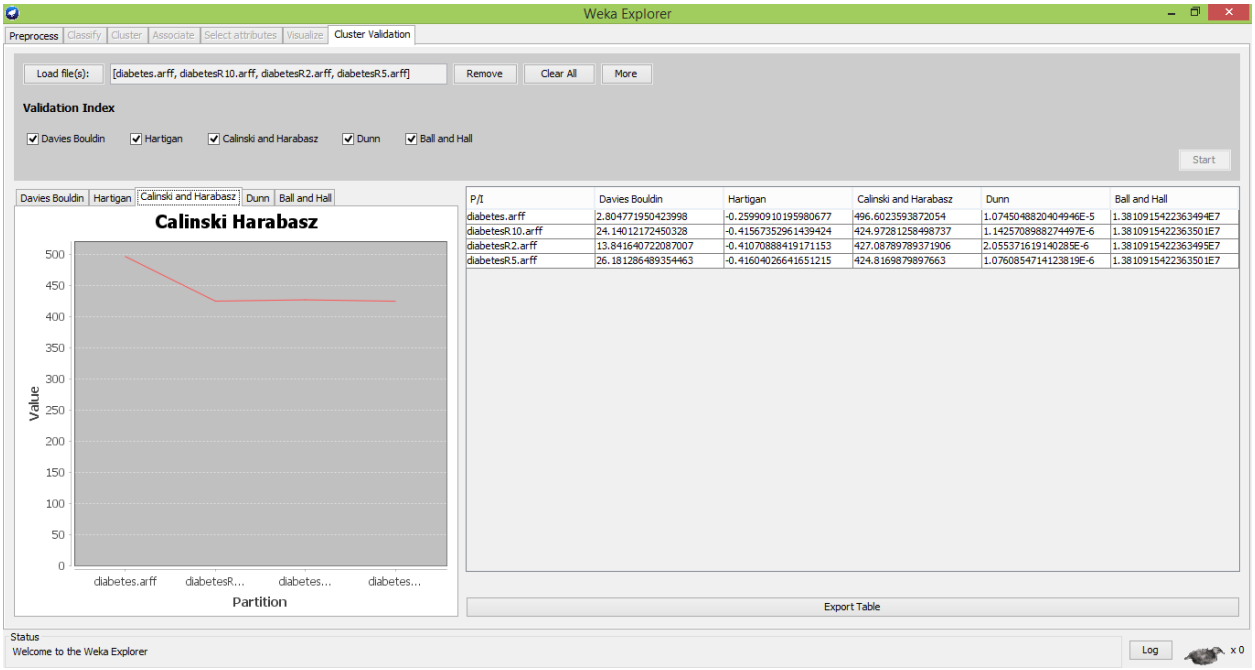


Figura 17. Resultados para *Diabetes*

Validación de la extensión de la Interfaz Gráfica de Usuario

A continuación se enuncian las validaciones realizadas a la extensión, se explica brevemente en que consiste y se muestra el error que aparece en la pantalla.

Cargar ficheros de agrupaciones que no pertenezcan a la misma base de datos.

Se muestra un mensaje de alerta para que el usuario sepa que los índices calculados para los ficheros importados pueden no tener sentido significativo o estar en correspondencia con la realidad.

Figura 18. Ficheros con agrupaciones de distintas bases de casos

Cargar ficheros erróneos

Se muestra un error de lectura del fichero con la descripción del error

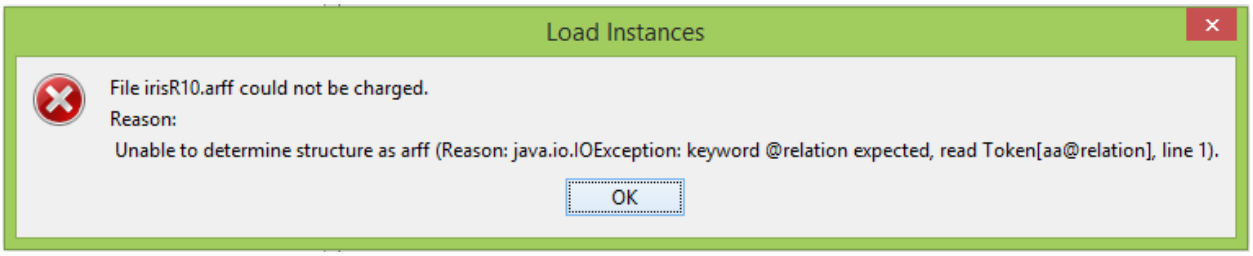


Figura 19. Error al intentar cargar un archivo dañado

Error al intentar exportar

Se muestra un error si se intenta exportar una tabla sin que esta tenga valores o haya sido creada

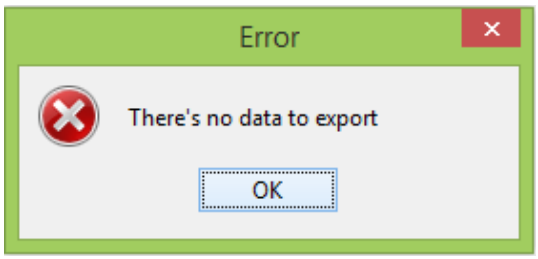


Figura 20. Error al intentar exportar

Conclusiones parciales

En este capítulo se describe la forma de validar las nuevas funcionalidades añadidas utilizando bases de conocimientos correctamente clasificadas y luego se desagruparon manualmente al azar. Luego se evaluaron los resultados de los índices para cada una de las bases arrojando estos los mejores resultados para las bases correctamente agrupadas. Es decir, si no se hubiese conocido de antemano la conformación de dichas agrupaciones el usuario hubiese escogido siempre la mejor.

Conclusiones generales

Al término del presente trabajo se puede afirmar que se cumplieron los objetivos trazados.

Se incorporaron nuevas funciones de distancia a Weka creando nuevas posibilidades de experimentación al aumentar la variedad de funciones para usar con el método de agrupamiento escogido. También le fueron añadidos índices de validación interna para comprobar la calidad de los agrupamientos y compararlos entre sí, ampliando de esta manera el marco de trabajo de la aplicación pues estos no formaban parte de la misma.

Fue comprobado el correcto funcionamiento de los índices de validación utilizando bases de conocimientos correctamente clasificadas y luego desagrupadas al azar, indicando los índices en cada caso los mejores resultados para las bases correctas.

Con estas nuevas características se enriquece la herramienta y la convierten en la idónea para complementar el trabajo “Procedimiento para la edición de conjuntos de entrenamiento en problemas no supervisados”, incrementando también su usabilidad.

Recomendaciones

A pesar de que se cumplieron los objetivos de este trabajo y se validó el mismo se propone:

- Añadirle nuevas funciones de distancia para ampliar las posibilidades de experimentación
- Añadirle otros índices de validación para hacer más potentes las comparaciones de los resultados
- Utilizar en la disciplina Inteligencia Artificial de la carrera Ingeniería Informática

Referencias

[1] David Hand, Heikki Mannila, and Padhraic Smyth, *Principles of Data Mining*. The MIT Press, 2001.

[2] J Hernández Orallo, M J Ramírez Quintana, and C Ferri Ramírez, *Introducción a la minería de datos*. .

[3] J. M. Molina and J. García, *Técnicas de análisis de datos aplicaciones prácticas utilizando Microsoft Excel y Weka*. 2006.

[4] Rui Xu and Donald C. Wunsch II, *Clustering*. New Jersey: A JOHN WILEY & SONS, INC., PUBLICATION, 2009.

[5] Garima Sehgal and Dr. Kanwal Garg, "Comparison of Various Clustering Algorithms," vol. Vol. 5 (3), 2014, pp. 3074–3076.

[6] K. -L. Du, "Clustering: A neural network approach," *Neural Netw.*, vol. 23, pp. 89–107, 2010.

[7] S. Pandit and S. Gupta, "A comparative study on distance measuring approaches for clustering," *Int. J. Res. Comput. Sci.*, pp. 29–31, 2011.

[8] Anna Huang, "Similarity Measures for Text Document Clustering," presented at the New Zealand Computer Science Research Student Conference., New Zealand, 2008.

[9] Rui Xu and Donald Wunsch II, "Survey of Clustering Algorithms," 2005, vol. 16.

[10] Amit Singhal, "Modern Information Retrieval: A Brief Overview." IEEE Computer Society Technical Committee on Data Engineering, 2001.

[11] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar, *Introduction to Data Mining*. Addison-Wesley, 2006.

[12] Joseph Lee Rodgers and W. Alan Nicewander, "Thirteen Ways to Look at the Correlation Coefficient," vol. 42, The American Statistician, 1988, pp. 59–66.

[13] Raffaele Giancarlo, Giosu`e Lo Bosco, and Luca Pinello, "Distance Functions, Clustering Algorithms and Microarray Data Analysis." 2010.

[14] Marggie D. González Toledo, *Una comparación de índices de validación de conglomerados*. Universidad de Puerto Rico, 2005.

[15] J Dunn, "Well separated clusters and optimal fuzzy partitions," *J. Cybern.*, vol. 4, pp. 95–104, 1974.

[16] T Calinski and J Harabasz, "A dendrite method for cluster analysis," *Commun. Stat. - Theory Methods*, vol. 3, pp. 1–27, 1974.

[17] G H Ball and D J Hall, "ISODATA, A novel method of data analysis an pattern classification," Stanford Research Institute, Menlo Park, 1965.

[18] Evgenia Dimitriadou, Sara Dolnicar, and Andreas Weingessel, "An examination of indexes for determining the number of clusters in binary data sets," *Psychometrika*, vol. 67, pp. 137–160, 2002.

[19] Remco R. Bouckaert, Eibe Frank, Mark Hall, Richard Kirkby, Peter Reutemann, Alex Seewald, and David Scuse, *WEKA Manual for Version 3-7-5*. 2011.

Bibliografía

[1]

“5 de los mejores software de minería de datos de Código Libre y Abierto | El rincón de JMACOE.” [Online]. Available: file:///C:/Users/Apolo/Downloads/herramientas%20de%20mineria%20de%20datos/5%20de%20los%20mejores%20software%20de%20miner%C3%ADa%20de%20datos%20de%20C%C3%B3digo%20Libre%20y%20Abierto%20_%20El%20rinc%C3%B3n%20de%20JMACOE.htm. [Accessed: 17-Dec-2014].

[2]

S. Pandit and S. Gupta, “A comparative study on distance measuring approaches for clustering,” *International Journal of Research in Computer Science*, pp. 29–31, 2011.

[3]

T. Caliński and J Harabasz, “A dendrite method for cluster analysis. Communications in Statistics,” vol. 3, Communications in Statistics, 1972, pp. 1–27.

[4]

E. Dimitriadou, Sara Dolnicar, and A. Weingessel, “An Examination of Indexes for Determining the Number of Clusters in Binary Data Sets,” 2002, pp. 137–160.

[5]

S. Sharma, *Applied multivariate techniques*. John Wiley & Sons Inc, 1996.

[6]

Rui Xu and Donald C. Wunsch II, *Clustering*. New Jersey: A JOHN WILEY & SONS, INC., PUBLICATION, 2009.

[7]

Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis, “Clustering Validity Checking Methods: Part II.”

[8]

Garima Sehgal and Dr. Kanwal Garg, “Comparison of Various Clustering Algorithms,” vol. Vol. 5 (3), 2014, pp. 3074–3076.

[9]

Florin Gorunescu, *Data Mining - Concepts, Models and Techniques*, vol. 12. 2011.

[10]

Raffaele Giancarlo, Giosu`e Lo Bosco, and Luca Pinello, “Distance Functions, Clustering Algorithms and Microarray Data Analysis.” 2010.

[11]

“Herramientas de Minería de Datos - Monografias.com.” [Online]. Available: file:///C:/Users/Apolo/Downloads/herramientas%20de%20mineria%20de%20datos/Herramientas%20de%20Miner%C3%ADa%20de%20Datos%20-%20Monografias.com.htm. [Accessed: 17-Dec-2014].

[12]

J Hernández Orallo, M J Ramírez Quintana, and C Ferri Ramírez, *Introducción a la minería de datos*. .

[13]

Ciro Rodríguez León, “Introducción Cap I V1.16.” 2014.

[14]

Geoffrey H. Ball and David J. Hall, “ISODATA, A novel method of data analysis an pattern classification,” 1965.

[15]

Amit Singhal, “Modern Information Retrieval: A Brief Overview.” IEEE Computer Society Technical Committee on Data Engineering, 2001.

[16]

M. HALKIDI, Y. BATISTAKIS, and M. VAZIRGIANNIS, “On clustering validation techniques,” Journal of Intelligent Information Systems, 2001, pp. 107–145.

[17]

Mtra. María Teresa Escobedo Portillo and PhD Jorge A. Salas Plata Mendoza, “P. CH. MAHALANOBIS Y LAS APLICACIONES DE SU DISTANCIA ESTADISTICA,” 2008.

[18]

David Hand, Heikki Mannila, and Padhraic Smyth, *Principles of Data Mining*. The MIT Press, 2001.

[19]

“Red neuronal artificial - Wikipedia, la enciclopedia libre.” [Online]. Available: http://es.wikipedia.org/wiki/Red_neuronal_artificial. [Accessed: 16-Dec-2014].

[20]

Anna Huang, “Similarity Measures for Text Document Clustering,” presented at the New Zealand Computer Science Research Student Conference., New Zealand, 2008.

[21]

Rui Xu and Donald Wunsch II, “Survey of Clustering Algorithms,” 2005, vol. 16.

[22]

J. M. Molina and J. García, *Técnicas de análisis de datos aplicaciones prácticas utilizando Microsoft Excel y Weka*. 2006.

[23]

Joseph Lee Rodgers and W. Alan Nicewander, “Thirteen Ways to Look at the Correlation Coefficient,” vol. 42, The American Statistician, 1988, pp. 59–66.

[24]

Marggie D. González Toledo, *Una comparación de índices de validación de conglomerados*. Universidad de Puerto Rico, 2005.

[25]

Yanchi Liu, Zhongmou Li, Hui Xiong, Xuedong Gao, and Junjie Wu, “Understanding of Internal Clustering Validation Measures,” presented at the IEEE International Conference on Data Mining, 2010.

[26]

“Weka (aprendizaje automático) - Wikipedia, la enciclopedia libre.” [Online]. Available: http://es.wikipedia.org/wiki/Weka_(aprendizaje_autom%C3%A1tico). [Accessed: 16-Dec-2014].

[27]

Remco R. Bouckaert, Eibe Frank, Mark Hall, Richard Kirkby, Peter Reutemann, Alex Seewald, and David Scuse, *WEKA Manual for Version 3-7-5*. 2011.

[28]

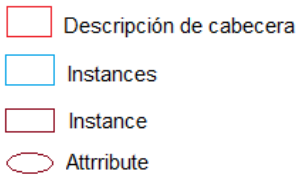
J. C. Dunn, “Well-Separated Clusters and Optimal Fuzzy Partitions,” vol. 4, Journal of Cybernetics, 1974, pp. 95–104.

Anexos

```
% 1. Title: Iris Plants Database
%
% 2. Sources:
%      (a) Creator: R.A. Fisher
%      (b) Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
%      (c) Date: July, 1988
%
@RELATION iris

@ATTRIBUTE sepallength  NUMERIC
@ATTRIBUTE sepalwidth   NUMERIC
@ATTRIBUTE petallength  NUMERIC
@ATTRIBUTE petalwidth   NUMERIC
@ATTRIBUTE class        {Iris-setosa,Iris-versicolor,Iris-virginica}
```

```
@DATA
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
```



Anexo 1. Ejemplo de fichero arff.

Fragmentos significativos de código fuente.

- Anexo 2: Implementar función de distancia Coseno

```
public double distance(Instance first, Instance second) {
    double sumTop = 0;
     double sumOne = 0;
     double sumTwo = 0;
     for (int i = 0; i < first.numAttributes(); i++) {
        sumTop += first.value(i) * second.value(i);
        sumOne += first.value(i) * first.value(i);
        sumTwo += second.value(i) * second.value(i);
    }
```

```
    double cosSim = sumTop / (Math.sqrt(sumOne) * Math.sqrt(sumTwo));
    if (cosSim < 0)
        cosSim = 0;//This should not happen, but does because of rounding errors
    return 1-cosSim;
  }
```

- Anexo 2: Implementar función de distancia Coeficiente de Correlación de Pearson

```
public double distance(Instance first, Instance second) {
        double xy = 0, x = 0, x2 = 0, y = 0, y2 = 0;
        for (int i = 0; i < first.numAttributes(); i++) {
          xy += first.value(i) * second.value(i);
          x += first.value(i);
          y += second.value(i);
          x2 += first.value(i) * first.value(i);
          y2 += second.value(i) * second.value(i);
        }
        int n = first.numAttributes();
        return 1 - ((xy - (x * y) / n) / Math.sqrt((x2 - (x * x) / n) * (y2 - (y * y) / n)));
      }
```

- Anexo 2: Clase Validation

```
public final class Validation {
  private int numClusters;
  private final Instances centroids;
  private final double[] SSWDB;
  private final int[] cMembers;
  private double SSB;
  private double SSWTotal;
  private final int numInstances;
  private final double[] SSW;
  private final int numAttributes;
  private final Instance globalMean;
```

```
public Validation(Instances data) throws Exception {
    data = removeIgnoreCols(data, "1");
    int[] assignments = getAssignments(data);
    centroids = getCentroids(data, assignments);
    data = removeIgnoreCols(data, "" + data.numAttributes() + "");
    numClusters = centroids.size();
    numInstances = data.size();
    numAttributes = data.numAttributes();

    SSW = new double[numClusters];
    SSB = 0;
    cMembers = new int[numClusters];
    SSWDB = new double[numClusters];

    globalMean = new DenseInstance(numAttributes);
    for (int i = 0; i < numAttributes; i++) {
        globalMean.setValue(i, data.meanOrMode(i));
    }

    for (int i = 0; i < numInstances; i++) {
        double d = distance(data.instance(i), centroids.instance(assignments[i]));
        SSW[assignments[i]] += (d * d);
        SSWDB[assignments[i]] += (d);
        cMembers[assignments[i]]++;
    }

    for (int i = 0; i < numClusters; i++) {
```

```
        double d = distance(centroids.instance(i), globalMean);
        SSB += cMembers[i] * (d * d);
    }

    for (int i = 0; i < numClusters; i++) {
        SSWTotal += SSW[i];
    }
}

public double getDunn() {

    //Calculating the maximum value of  SSW
    double maxSSW = 0;

    for (int i = 0; i < numClusters; i++) {
        double actual = SSW[i];
        if (actual > maxSSW) {
            maxSSW = actual;
        }
    }

    Matrix R = new Matrix(numClusters, numClusters);
    for (int i = 0; i < (numClusters - 1); i++) {
        for (int j = i + 1; j < numClusters; j++) {
            double d = distance(centroids.instance(i), centroids.instance(j)) / maxSSW;
            R.set(i, j, d);
            R.set(j, i, d);
        }
```

```
    }

    double[] minS = new double[numClusters - 1];
    for (int i = 0; i < numClusters - 1; i++) {
        double[] S = new double[numClusters - i - 1];
        int k = 0;
        for (int j = i + 1; j < (numClusters); j++) {
            S[k] = R.get(i, j);
            k++;
        }
        minS[i] = min(S);
    }
    double Dunn = min(minS);
    //If the partition contains compact and well-separated clusters, Dunn's Index will be large,
    //since the distance between clusters is expected to be large and the diameter of the
    //cluster is expected to be small.
    return Dunn;
}

public double getCalinskiHarabans() {

    double CH;

    CH = SSB / (numClusters - 1);
    CH = CH / (SSWTotal / (numInstances - numClusters));

    return CH;
}
```

```
    public double getBallHall() {
        double BH;
        BH = SSWTotal / numClusters;

        return BH;
    }

    public double getDaviesBouldin() {

        Matrix Rij = new Matrix(numClusters, numClusters);
        for (int i = 0; i < (numClusters - 1); i++) {
            for (int j = i + 1; j < numClusters; j++) {
                double d = (SSWDB[i] / cMembers[i] + SSWDB[j] / cMembers[j]) /
distance(centroids.instance(i), centroids.instance(j));
                Rij.set(i, j, d);
                Rij.set(j, i, d);
            }
        }

        double[] maxRi = new double[numClusters];

        double DB = 0;
        System.out.println("Matrix Rij Validacion1 " + Rij);
        for (int i = 0; i < numClusters; i++) {
            maxRi[i] = -1;
            for (int j = 0; j < numClusters; j++) {
                double actual = Rij.get(i, j);
```

```
            if (actual > maxRi[i]) {
                maxRi[i] = actual;
            }
        }
        DB += maxRi[i];
    }
    //Small values of DB are indicative of the presence of compact and well-separated clusters.
    DB = DB / numClusters;

    return DB;
}
public double getHartigan() {
    double Hartigan;
    Hartigan = log(SSB / SSWTotal);
    return Hartigan;
}
private double distance(Instance x, Instance y) {
    double sum = 0;
    for (int i = 0; i < x.numAttributes(); i++) {
        double valX = x.value(i);
        double valY = y.value(i);
        //ignore missing values
        if (!Double.isNaN(valX) || Double.isNaN(valY)) {
            sum += (valY - valX) * (valY - valX);
        }
    }
    return Math.sqrt(sum);
}
```

```
private double min(double[] S) {
    double min = Double.MAX_VALUE;
    for (int i = 0; i < S.length; i++) {
        if (S[i] != 0) {
            min = (min > S[i]) ? S[i] : min;
        }
    }
    return min;
}
private double max(double[] S) {
    double max = Double.MIN_VALUE;
    for (int i = 0; i < S.length; i++) {
        max = (max > S[i]) ? max : S[i];
    }
    return max;
}

//Looking for the centroids
private Instances getCentroids(Instances data, int[] assignments) throws Exception {

    double[] m_clusterAssignments;

    //Determining the quantity of clusters
    numClusters = data.attribute("Cluster").numValues();
    data = removeIgnoreCols(data, "" + data.numAttributes() + "");

    int[] labels = new int[numClusters];
    Instance[] tempCentroides;
```

```
tempCentroides = new Instance[numClusters];

int cantAtrib = data.numAttributes();

// Initializing to zero the centroids
for (int i = 0; i < numClusters; i++) {
   tempCentroides[i] = new DenseInstance(cantAtrib);
   for (int k = 0; k < cantAtrib; k++) {
      tempCentroides[i].setValue(k, 0);
   }
}
int i = 0;
for (Instance dato : data) {
   int ci = assignments[i];
   labels[ci]++;

   // Updating the sum of for each centroid.
   for (int k = 0; k < cantAtrib; k++) {
      double a = tempCentroides[ci].value(k) + dato.value(k);
      if (!Double.isNaN(a)) {
         tempCentroides[ci].setValue(k, a + dato.value(k));
      }
   }
   i++;
}

// It splits for the amount of elements of the cluster.
for (int j = 0; j < numClusters; j++) {
```

```
            for (int k = 0; k < tempCentroides[j].numAttributes(); k++) {
                tempCentroides[j].setValue(k, tempCentroides[j].value(k) / labels[j]);
            }
        }

        Instances crt = new Instances(data);
        crt.delete();
        crt.addAll(Arrays.asList(tempCentroides));
        return crt;
    }

    private int[] getAssignments(Instances data) {
        int[] assignments = new int[data.size()];
        for (int i = 0; i < data.size() - 1; i++) {
            Instance inst = data.get(i);
            assignments[i] = (int) inst.value((inst.numAttributes() - 1));
        }
        return assignments;
    }

    private static Instances removeIgnoreCols(Instances inst, String cols) {

        Remove af = new Remove();
        Instances retI = null;

        try {
            af.setAttributeIndices(cols);
            af.setInvertSelection(false);
```

```
            af.setInputFormat(inst);
            retI = Filter.useFilter(inst, af);
        } catch (Exception e) {
        }

        return retI;
    }
}
```