RASEMUS: UNA HERRAMIENTA PARA EL DESCUBRIMIENTO DE CONOCIMIENTO EN BASES DE DATOS CON TÉCNICAS DE CLUSTERING, DÉBILMENTE ACOPLADO CON EL SGBD POSTGRESQL.

HERMES RICARDO NARVAEZ TERAN

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERIA
PROGRAMA DE INGENIERIA DE SISTEMAS
SAN JUAN DE PASTO
2009

RASEMUS: UNA HERRAMIENTA PARA EL DESCUBRIMIENTO DE CONOCIMIENTO EN BASES DE DATOS CON TÉCNICAS DE CLUSTERING, DÉBILMENTE ACOPLADO CON EL SGBD POSTGRESQL.

HERMES RICARDO NARVAEZ TERAN

Trabajo de Grado presentado como requisito para optar al título de
Ingeniero de Sistemas

Dr. Ricardo Timarán Pereira, Ph.D.
Director

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERIA
PROGRAMA DE INGENIERIA DE SISTEMAS
SAN JUAN DE PASTO
2009

“Las ideas y conclusiones aportadas en el trabajo de grado son responsabilidad exclusiva de sus autores”.

Artículo 1º. Del acuerdo No. 324 del 11 de Octubre de 1966 emanado del honorable Consejo Directivo de la Universidad de Nariño.

Nota de aceptación:

_____
_____
_____
_____
_____
_____

_____

Firma del presidente del jurado

_____

Firma del jurado

_____

Firma del jurado

San Juan de Pasto, Septiembre 8 de 2009

RESUMEN

EN ESTE PROYECTO DE INVESTIGACIÓN SE PRESENTA EL ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE "*RASEMUS: UNA HERRAMIENTA PARA EL DESCUBRIMIENTO DE CONOCIMIENTO EN BASES DE DATOS CON TÉCNICAS DE CLUSTERING, DÉBILMENTE ACOPLADO CON EL SGBD POSTGRESQL*", COMO LAS PRUEBAS DE FUNCIONALIDAD CON CONJUNTOS DE DATOS REALES Y ARTIFICIALES.

LA ARQUITECTURA DE RASEMUS ESTÁ COMPUESTA POR TRES MODULOS: EL MODULO DE KERNEL, EL MODULO DE INTERFAZ GRAFICA Y EL MÓDULO DE UTILIDADES

EL MODULO DE INTERFAZ GRAFICA PERMITE HACER UN MANEJO FACIL Y FUNCIONAL, EL MÓDULO DE KERNEL ES EL NÚCLEO DE RASEMUS QUE PERMITE LA EJECUCIÓN DE LOS ALGORITMOS DE CLUSTERING DE TIPO PARTICIONAL, JERARQUICO Y BASADO EN DENSIDAD. TAMBIEN PERMITE LA EJECUCION DE LOS DIFERENTES FILTROS Y LA ESTRUCTURACION DE LOS RESULTADOS Y FINALMENTE EL MÓDULO DE UTILIDADES QUE ENTRE SUS OPERACIÓNES ESTAN, PERMITIR LA CONEXIÓN CON EL SGBD POSTGRESQL, LA CONEXIÓN A ARCHIVOS, LA EXPORTACION DE ARCHIVOS CON FORMATO ENTRE OTRAS.

ABSTRACT

IN THIS RESEARCH PROJECT PRESENTS AN ANALYSIS, DESIGN AND IMPLEMENTATION OF "RASEMUS: A TOOL FOR KNOWLEDGE DISCOVERY IN DATABASES CLUSTERING TECHNIQUES, LOOSELY COUPLED WITH THE DBMS POSTGRESQL" AS TEST DATA SETS WITH FUNCTIONALITY REAL AND ARTIFICIAL.

THE ARCHITECTURE OF RASEMUS IS COMPOSED OF THREE MODULES: THE KERNEL MODULE, THE MODULE GUI AND UTILITIES MODULE

MODULE GUI ALLOWS EASY HANDLING AND FUNCTIONALITY, THE KERNEL MODULE IS THE CORE OF RASEMUS PERMITTING THE ENFORCEMENT OF CLUSTERING ALGORITHMS TYPE PARTICIONAL, HIERARCHICAL AND DENSITY BASED. IT ALSO ALLOWS THE IMPLEMENTATION OF THE VARIOUS FILTERS AND STRUCTURING OF RESULTS AND FINALLY THE UTILITY MODULE FROM ITS OPERATIONS ARE TO ALLOW CONNECTION WITH DBMS POSTGRESQL, THE CONNECTION TO ARCHIVES OF EXPORT FORMAT FILES AMONG OTHERS.

ÌNDICE GENERAL

ÍNDICE DE FIGURAS

pag.

ÍNDICE DE TABLAS

ÍNDICE DE ANEXOS

pag.

GLOSARIO

DISTANCIA: La distancia expresa la proximidad o lejanía entre dos objetos.

SIMILITUD: Relación de semejanza o parecido entre objetos.

MEDIA: o media aritmética (también llamada promedio), de un conjunto finito de números, es igual a la suma de todos sus valores dividida entre el número de sumandos.

MODA: es el valor con una mayor frecuencia en una distribución de datos.

FRECUENCIA: En estadística, se llama frecuencia a la cantidad de veces que se repite un determinado valor de la variable.

FRECUENCIA ABSOLUTA (N_i): la frecuencia absoluta de una variable estadística X_i, es el número de veces que aparece en el estudio este valor. A mayor tamaño de la muestra, aumentará el tamaño de la frecuencia absoluta; es decir, la suma total de todas las frecuencias absolutas debe dar el total de la muestra estudiada (N).

FRECUENCIA RELATIVA (f_i): es el cociente entre la frecuencia absoluta y el tamaño de la muestra (N).

INTRODUCCION

El incremento del volumen de información en todo tipo de organizaciones aumenta de una manera considerable cada día. Todo este volumen de información histórica aparte de su función de "memoria de la organización" pude ser útil para explicar el pasado, el presente y predecir la información futura [34] utilizando Herramientas orientadas a *Business Intelligence o Inteligencia de Negocios*.

El Proceso de extraer conocimiento a partir de grandes volúmenes de datos ha sido reconocido por muchos investigadores como tópico de investigación clave en los sistemas de bases de datos, y por muchas compañías industriales como una importante área y una oportunidad para obtener mayores ganancias[34].

El Descubrimiento de Conocimiento en Bases de Datos (DCBD), es básicamente un proceso automático en el que se combina descubrimiento y análisis. El proceso consiste en extraer patrones de reglas o funciones, a partir de los datos, para que el usuario los analice. Esta tarea implica generalmente preprocesar los datos, aplicar técnicas de minería de datos (Data Mining) y presentar resultados [19] [41] [59]. Una de las técnicas más importante de la minería de datos es la agrupación o clasificación no supervisada más conocida como Clustering. Su objetivo es particionar los datos obteniendo el conocimiento de acuerdo a las características de los mismos. Los objetos son agrupados basándose en el principio de maximización de similitud dentro de los clústeres y minimización de similitud entre clústeres diferentes, para ello utilizan funciones de distancia y similitud o disimilitud dependiendo el tipo de datos a trabajar, ya sean de tipo numéricos o categóricos respectivamente[32][52].Los algoritmos de clustering están clasificados en técnicas dependiendo de la forma como se hace la tarea de agrupación, entre ellas encontramos las técnicas de clustering jerárquico, particional, basados en densidad, basados en grid o grillas, difuso, de redes neuronales artificiales, con algoritmos evolutivos y otras más [33][52]. Hay que resaltar que el Clustering puede ser una tarea inicial de todo proyecto de Data Mining ya que sirve de soporte a todas las actividades que estos involucran [14] [52].

Una herramienta DCBD puede clasificarse dependiendo de su arquitectura en herramientas débilmente acopladas, medianamente acopladas o fuertemente acopladas con el sistema gestor de bases de datos (SGDB) de la forma como estén interactuando estos dos elementos y como fueron implementadas las tareas de descubrimiento de conocimiento [65]. Además de las técnicas en mención, debe estar dotada de elementos de selección, transformación y visualización que le faciliten al analista su labor.

TÍTULO

RASEMUS: Una herramienta para el descubrimiento de conocimiento en Bases de Datos con técnicas de clustering, débilmente acoplada con el SGBD PostgreSQL.

LÍNEA DE INVESTIGACIÓN

El presente trabajo de grado, se encuentra inscrito bajo la línea de software y manejo de información. Se marca dentro del área de las Bases de Datos, y específicamente en la subárea de arquitecturas de integración del Proceso de Descubrimiento en Bases de Datos con Sistemas Gestores de Bases de Datos.

ALCANCE Y DELIMITACIONES

RASEMUS es una herramienta que contempla todas las etapas del proceso de DCBD, es decir: Selección, preprocesamiento, transformación, minería de datos y visualización [13][34][53] La etapa de minería de datos está enfocada a las técnicas de agrupación o segmentación o clasificación no supervisada denominada clustering [13][34][50][52]. Se estudio las diferentes técnicas de clustering [13] [31] [32] [33] [52] y sus correspondientes algoritmos y aplicaciones [13] [21] y se implementaron las más utilizadas. Se contemplo el desarrollo de una interfaz gráfica que le permite al usuario interactuar de una manera fácil con la herramienta.

Para los algoritmos implementados, se hicieron pruebas de funcionalidad utilizando conjuntos de datos reales y repositorios existentes en la web [86], analizando y evaluando los resultados obtenidos.

PROBLEMA OBJETO DE ESTUDIO

El aumento de información digital en las instituciones de todo el mundo y de todo tipo, ha hecho que muchos investigadores generen técnicas de minería para aprovecharla al máximo.

Dentro de la minería de datos las técnicas más utilizadas son las de agrupación o segmentación o clasificación no supervisado o de Clustering [16] [33] [52] [54]. Además dentro las técnicas de Clustering también se pueden encontrar varias clasificaciones de ellas dependiendo de su aplicación y su forma de agrupación.

Cuando se analizan grandes volúmenes de datos las herramientas tradicionales tales como hojas de cálculo son insuficientes e inapropiadas por su bajo rendimiento. Además las herramientas existentes el mercado que contemplan las

técnicas de clustering tales como Clementine, SPSS [68], DBMiner [73], Intelligent Miner [82], entre otras, necesitan de la adquisición de costosas licencias. Hecho que limita a las pequeñas, medianas empresas, grupos de investigación y organizaciones públicas o privadas la utilización de este tipo de herramientas para soportar sus decisiones que puede conllevar a pérdidas económicas y de competitividad.

Por otra parte herramientas tales como Weka [100], Tariy [99], Orange [91], AlfhaMiner [70], desarrolladas bajo software libre no contemplan la etapa de clustering o se han implementado únicamente algoritmos de un tipo de clustering.

En esta propuesta se plantea desarrollar una herramienta especializada para la tarea de clustering que incluya varias técnicas y diferentes algoritmos, bajo software libre, que permita que las pequeñas, medianas empresas, grupos de investigación y organizaciones públicas o privadas cuenten con una herramienta libre para soportar sus decisiones.

OBJETIVOS

Objetivo General.

desarrollo de una herramienta para el descubrimiento de conocimiento en Bases de Datos con técnicas de Clustering, débilmente acopladas con el SGBD PostgreSQL bajo Software Libre para facilitar el proceso de toma de decisiones a las pequeñas, medianas empresas, grupos de investigación y organizaciones públicas y privadas.

Objetivos Específicos.

Recopilar información acerca de las diferentes técnicas de clustering para minería de datos.

Analizar las diferentes técnicas y algoritmos de clustering para minería de datos.

Evaluar diferentes herramientas DCBD débilmente acopladas con un SGDB que implementen técnicas de Clustering.

Analizar, diseñar y desarrollar programas que permitan la selección, preprocesamiento y la transformación de datos.

Escoger los algoritmos de clustering más apropiados para determinadas aplicaciones e implementarlos en un programa.

Realizar pruebas de funcionalidad y pruebas con conjuntos de datos reales.

Dar a conocer los resultados del proyecto a través de la publicación de un artículo en una revista o conferencia nacional o internacional.

JUSTIFICACIÓN

Todas las herramientas de DCBD o comúnmente conocidas como herramientas de minería de datos sirven en todo tipo de organización para apoyar la toma de decisiones de forma semiestructurada y agilizan el proceso de extracción de conocimiento, que muchas veces no es fácil de encontrar. Es claro que el diseño de estas herramientas provee de una mayor capacidad de análisis que otras comúnmente utilizadas como las hojas de cálculo. Están construidas explícitamente con diversos modelos para obtener patrones insospechados a partir de datos. Apoyan la toma de decisiones al permitir al analista extraer información útil que antes se encontraba en cúmulos gigantes de datos. Diversas herramientas de minería de datos están disponibles en el mercado, estas ofrecen diferentes tipos de arquitecturas que determinan, en alguna medida, su versatilidad y su costo. Por lo general todas estas son demasiado costosas, necesitan de licenciamiento para su uso, además es importante tener en cuenta que el constante trabajo investigativo en el mundo en cuestiones de algoritmos de clustering cada día ha ido evolucionando y se hace necesario tenerlos en cuenta para buscar nuevos campos de aplicación de los mismos y mejorar la extracción de conocimiento. Muchas de estas herramientas solo abordan algoritmos de clustering de un solo tipo, lo que hace que no se aprovechen al máximo este tipo de técnicas ya que el clustering puede ser una tarea inicial de todo proyecto de minería de datos dando soporte a todas las actividades como clasificación o asociación y hasta de los mismos algoritmos de clustering, que muchas veces no satisfacen los resultados esperados.

El desarrollo de *RASEMUS*, como una herramienta DCBD bajo licencia publica GNU, permitirá que empresas, grupos de investigación u organización que por su tamaño no puedan acceder a estas herramientas de carácter propietario, utilicen esta tecnología para mejorar la toma de decisiones, maximizar sus ganancias con decisiones acertadas, eleven su poder competitivo y den soporte a sus investigaciones.

Por ser *RASEMUS* una herramienta genérica de DCDB, puede utilizarse en diferentes campos como la industria, la banca, la salud, educación, psicología y muchos otros que requieran técnicas de Clustering.

Por otra parte, *RASEMUS* se convierte en otro aporte más en el área del Descubrimiento de Conocimiento en Bases de Datos, que la Universidad de Nariño, a través de su programa de Ingeniería de Sistemas, contribuye a la investigación científica y al desarrollo de la región, el país y el resto del mundo.

ORGANIZACIÓN DEL DOCUMENTO

Este documento se encuentra organizado de la siguiente manera: en la primera parte se especifican el objetivo general y objetivos específicos del proyecto, en la capitulo 1 se presenta todo lo concerniente al DCBD y sus etapas, en el capítulo 2 se presenta todo los conceptos más importantes de la tarea clustering, comenzado por su definición, las características que deben tener los algoritmos, pasando por formas de medir distancia en espacios multidimensionales, presentado las diferentes técnicas de clustering y finalizando con las diferentes aplicaciones en los cuales han sido utilizadas este tipo de técnicas. El en capitulo 3 se desarrollan los algoritmos implementados en RASEMUS, entre los cuales se encuentran algoritmos propuestos en el laboratorio GRIAS de la línea KDD denominados WayCluster y el algoritmo K-Frequency que es un algoritmo basado en el algoritmo K-Histogram. En el capítulo 4 se muestra el análisis de algunas herramientas que tienen implementadas tareas de clustering. En el capítulo 5 se presentan las herramientas que se utilizaron en el desarrollo de RASEMUS. En el capítulo 6 se presenta el análisis, diseño y modelado de RASEMUS. El en capítulo 7 se presenta la forma como fue implementación del proyecto. En el capítulo 8 se muestran las pruebas y los resultados, en el capítulo 9 están las conclusiones y en capitulo 10 están las recomendaciones seguido por las referencias bibliográficas y los anexos.

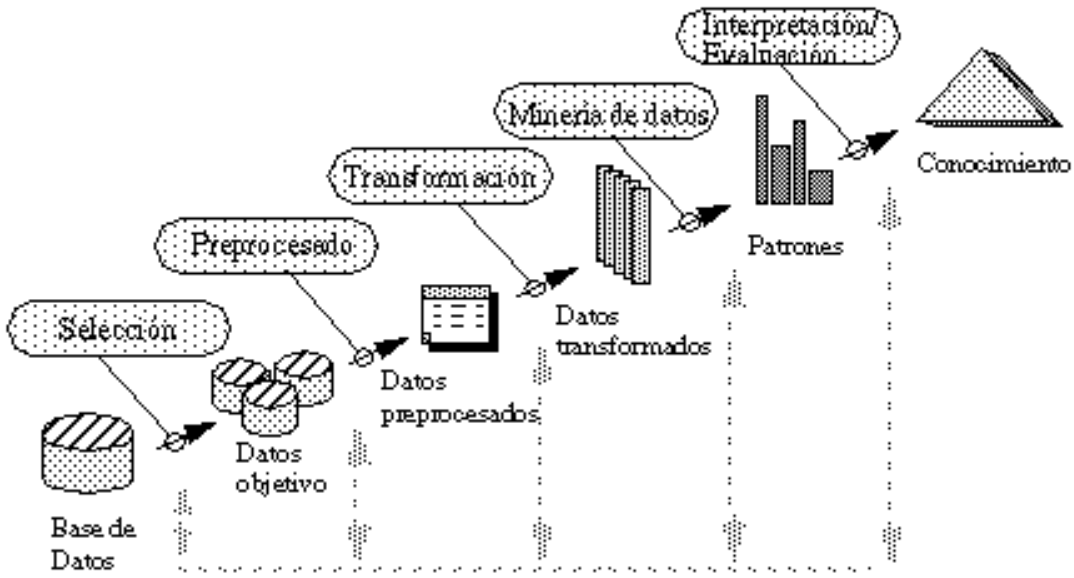
1. EL PROCESO DE DESCUBRIMIENTO DE CONOCIMIENTO EN BASES DE DATOS (DCBD)

El proceso de DCBD es el proceso que utiliza técnicas de minería de datos para extraer o identificar patrones de información para ser evaluados e interpretados, de acuerdo a las especificaciones de medias y umbrales, usando una base de datos con alguna selección, preprocesamiento, muestreo y transformación, con el fin de obtener lo que se piensa es conocimiento para soportar la toma de decisiones [23] [34].

1.1 ETAPAS DEL PROCESO DCBD

El proceso DCBD es interactivo e iterativo, involucra numerosos pasos con la intervención del analista en la toma de muchas decisiones y se resumen en las etapas de selección, preprocesamiento, transformación, minería de datos e interpretación o evaluación de resultados [79] (ver figura 1):

Figura 1: Etapas del proceso DCBD



1.1.1 Etapa de selección.

En la etapa de Selección, una vez identificado el conocimiento relevante y prioritario y definidas las metas del proceso DCBD, desde el punto de vista del usuario final, se crea un conjunto de datos objetivo, seleccionando todo el conjunto de datos o una muestra representativa de esos datos, sobre los cuales se va a realizar el proceso de descubrimiento. La selección de los datos varía de acuerdo con los objetivos del negocio.

1.1.2 Etapa de preprocesamiento/data cleaning

En la etapa de Preprocesamiento/Data cleaning se analiza la calidad de los datos, se aplican operaciones básicas como la remoción de datos ruidosos y selección de estrategias para el manejo de datos desconocidos (missing y empty), datos nulos, datos duplicados y uso de técnicas estadísticas para el reemplazo. En esta etapa es de suma importancia la interacción con el usuario o analista.

Los datos ruidosos (noisy data) son valores que están significativamente fuera del rango de valores esperado. Se debe principalmente por errores humanos, por cambios en el sistema, por la información no disponible a tiempo y por las fuentes heterogéneas de datos. Los datos desconocidos empty son aquellos que no les corresponde un valor en el mundo real (ejemplo., cesárea en los hombres) y los datos missing son aquellos que tienen un valor que no fue capturado (ejemplo, el número telefónico). Los datos nulos son datos desconocidos que son permitidos por los manejadores de bases de datos relacionales (RDBMS). En el proceso de limpieza todos estos valores se ignoran, se reemplazan por defecto, por el valor más cercano o se usan métricas de tipo estadístico como la media, mínimo, máximo, y de más

1.1.3 Etapa de Transformación/Reducción

En la etapa de transformación/reducción de datos, se buscan características útiles para representar los datos dependiendo de la meta del proceso. Se utilizan métodos de reducción de dimensiones o de transformación para disminuir el número efectivo de variables bajo consideración o para encontrar representaciones invariantes de los datos [24].

Los métodos de reducción de dimensiones pueden simplificar una tabla de una base de datos horizontalmente o verticalmente. La reducción horizontal implica la eliminación de tuplas idénticas como producto de la sustitución del valor de un atributo por otro de alto nivel en una jerarquía definida de valores categóricos o por la discretización de valores continuos (por ejemplo la edad por un rango de edades) o también cambiar un atributo con valores categóricos a valores numéricos. La reducción vertical implica la eliminación de atributos que son insignificantes o redundantes con respecto al problema como la eliminación de llaves, la eliminación de columnas que dependen funcionalmente (por ejemplo edad y fecha de nacimiento).

1.1.4 Etapa de minería de datos

El objetivo de la etapa data mining es la búsqueda y descubrimiento de patrones insospechados y de interés (por ejemplo el 80% de los hombres que compran cerveza también compran pañales desechables) utilizando técnicas de descubrimiento tales como clasificación [56][19], clustering [32][12][19], patrones secuenciales [5] y asociaciones [5] [4] [62] entre otras.

La escogencia de un algoritmo de minería de datos incluye la selección de los métodos para ser usados en la búsqueda de patrones en los datos, así como decidir cuales modelos y parámetros pueden ser apropiados de acuerdo al tipo de datos ya sean de tipo categórico o numérico y escoger un método particular de minería de datos con el criterio general del proceso DCBD (por ejemplo, el usuario final puede estar más interesado en entender el modelo, que sus capacidades predicativas).

Técnicas de minería de datos

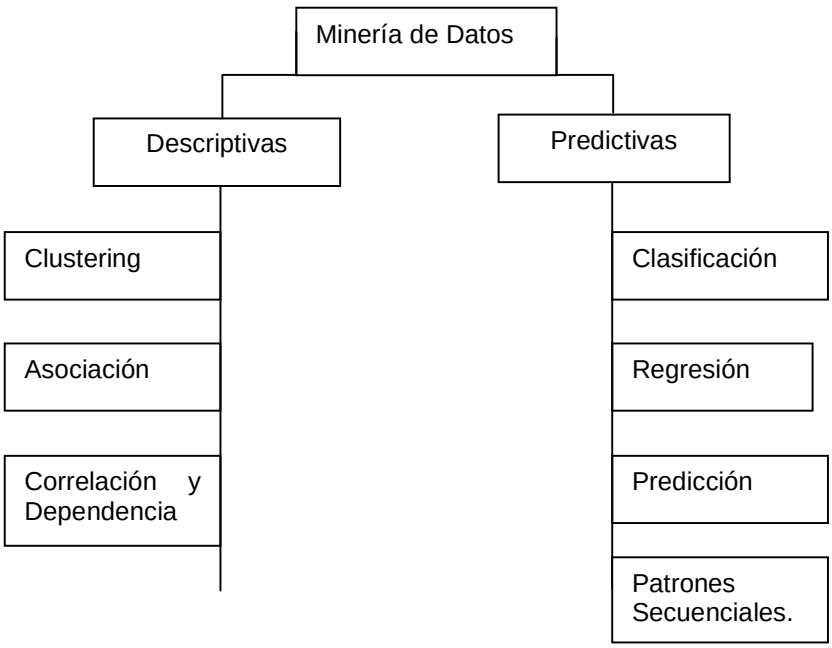
Las dos principales metas de la minería de datos son [25]: la verificación y el descubrimiento. En la primera el sistema se limita a verificar una hipótesis del usuario. En la segunda, el sistema automáticamente encuentra nuevos patrones. La meta de descubrimiento, a su vez, se subdivide en predicción y descripción.

En la predicción, el sistema involucra el uso de algunas variables o campos de una base de datos para predecir valores futuros o desconocidos de otras variables de interés [23]. En la descripción, el sistema encuentra patrones con el propósito de

presentarlos al usuario en una forma entendible, hace un barrido a través de la base de datos e identifica patrones escondidos.

Entre las técnicas de descripción se encuentran los métodos de clustering, asociación y correlación y dependencia, y entre los métodos de predicción se encuentran los métodos de clasificación, regresión, predicción. (Ver figura 2).

Figura 2: Clasificación de las técnicas de Minería de Datos.



Clustering: El proceso de agrupar objetos físicos o abstractos en clases de objetos similares se llama clustering o clasificación no supervisada [19]. Básicamente, el clustering agrupa un conjunto de datos (sin un atributo de clase predefinido) basado en el principio: maximizar la similitud intraclase y minimizar la similitud interclase. El análisis de clustering ayuda a construir particiones significativas de un gran conjunto de objetos basado en la metodología "divide y conquista" la cual descompone un sistema de escala grande en pequeños componentes para simplificar el diseño y la implementación.

El clustering ha sido estudiado por las áreas de la estadística, aprendizaje de máquina, bases de datos espaciales y minería de datos con diferentes énfasis.

El clustering se utiliza en el análisis de flujo de efectivo para grupo de clientes que pagan en un periodo del mes en particular (por ejemplo, cuando se recibe el cheque del seguro social o se deposita el cheque quincenal de nómina en la cuenta), la segmentación del mercado y para descubrir grupos de afinidades. También el clustering se utiliza para el descubrimiento de subpoblaciones homogéneas de consumidores en bases de datos de marketing.

Asociación: Es el proceso de descubrir relaciones o asociaciones entre los datos.

Generalmente, estas reglas se buscan en las llamadas bases de datos transaccionales. Como ejemplo de aplicación de este tipo de algoritmos, se tiene el caso del análisis de la canasta familiar. En estas bases de datos, los nombres de los atributos son nombres de productos, cuyo valor es 0, si no está presente en la transacción, o 1 si está presente. De este modo, cada fila representa una "compra" determinada del cliente, y se trata de buscar relaciones entre los objetos de la transacción, es decir, siempre que se compra A, se compra B. Por ejemplo siempre que se compra Arroz se compra Aceite.

Correlación o dependencia: Una dependencia o correlación funcional (aproximada o absoluta) se da cuando un atributo de un conjunto de datos determina valor de otro atributo. Un ejemplo es el siguiente caso, una persona ingresa en maternidad por consiguiente se deduce que esa persona es una mujer.

Clasificación: La clasificación de datos es el proceso por medio del cual se encuentra propiedades comunes entre un conjunto de objetos en una base de datos y se los clasifica en diferentes clases, de acuerdo al modelo de clasificación. Para construir un modelo de clasificación, una muestra E de la base de datos W se trata como el conjunto de entrenamiento, en el cual cada tupla se compone del mismo conjunto de atributos (o características) múltiples como las tuplas en la gran base de datos W y adicionalmente, cada tupla tiene una identidad de clase conocida (etiqueta) asociada.

El objetivo de la clasificación es primero analizar los datos entrenados y desarrollar una descripción exacta o un modelo para cada clase usando las características presentes en los datos. Así las descripciones de las clases se usan para clasificar futuros datos de prueba en la base de datos W o para desarrollar una mejor descripción (llamadas reglas de clasificación) por cada clase en la base de datos.

Entre los modelos de clasificación están: el modelo basado en la teoría Rough set [63], árboles de decisión [3][19], aprendizaje basado en ejemplos, aprendizaje basado en instancias.

Regresión: La regresión es la técnica más antigua y más conocida de las técnicas estadísticas de datos que se utiliza. Básicamente, la regresión tiene un conjunto que desarrolla fórmulas o modelos que se ajuste a los datos. Los resultados buscan predecir el comportamiento futuro, sólo tiene que tomar sus nuevos datos, aplicar la fórmula desarrollada y encontrar una predicción. La principal limitación de esta técnica es que sólo funciona bien con datos cuantitativos continuos (como el peso, la velocidad o la edad). [33]

Predicción: Encuentra una clasificación de valores faltantes o sin conocimiento previo. Se refiere tanto a la predicción de valores en los datos como a la predicción de clases utilizando la identificación de distribuciones en los datos disponibles. Un ejemplo de su utilización esta en predecir qué gobernador será elegido de acuerdo a las actuales encuestas electorales. [33]

Patrones Secuenciales: Consiste en descubrir relaciones entre los datos, que se recogen durante un periodo. De este modo, se puede determinar si dado un suceso A, ocurrirá un suceso B después de n meses. Básicamente se trata de una variante de la consulta de asociación en las cuales el factor tiempo interviene de forma relevante.

1.1.5 Etapa de interpretación / evaluación

En la etapa de interpretación/evaluación se interpretan los patrones descubiertos y posiblemente se retorna a los anteriores pasos o etapas para posteriores iteraciones. Esta etapa puede incluir la visualización de los patrones extraídos, la remoción de los patrones redundantes o irrelevantes y la traducción de los patrones útiles en términos que sean entendibles para el usuario. Por otra parte se consolida el conocimiento descubierto para incorporarlo en otro sistema para posteriores acciones, o simplemente documentarlo y reportarlo a las partes interesadas como también para verificar y resolver conflictos potenciales con el conocimiento previamente descubierto.

1.2 ARQUITECTURAS DE INTEGRACIÓN DE LAS HERRAMIENTAS DCBD CON UN SISTEMA GESTOR DE BASES DE DATOS (SGBD)

Las arquitecturas de integración DCBD con un SGBD se pueden ubicar en una de tres tipos: Herramientas débilmente acopladas, medianamente acopladas y fuertemente acopladas con un SGBD [65].

1.2.1 Arquitecturas débilmente acopladas

Es cuando los componentes de la herramienta se encuentran en una capa externa al SGBD, por fuera del núcleo y su integración con este se hace a partir de un interfaz [65].

1.2.2 Arquitecturas medianamente acopladas

Es cuando dentro de la herramienta se realizan tareas de descubrimiento de patrones que hacen parte del SGBD como funciones definidas por el usuario o procedimientos almacenados [65].

1.2.3 Arquitecturas fuertemente acopladas

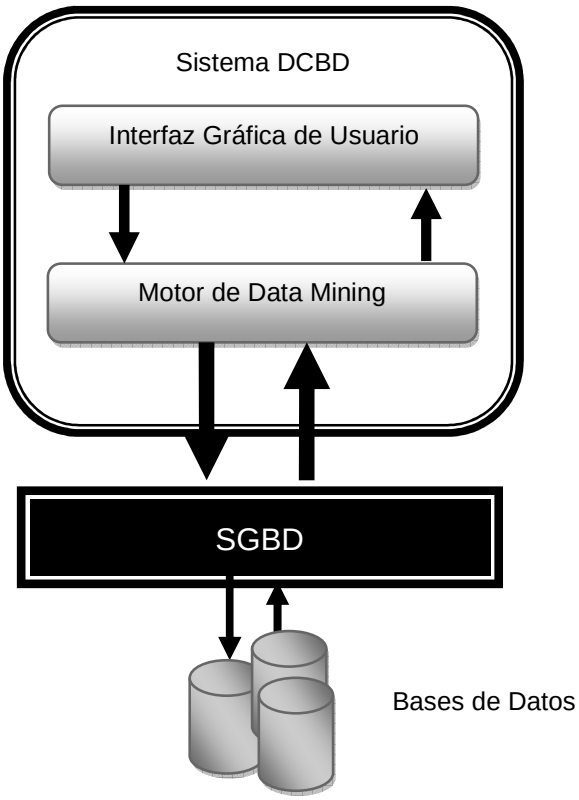
Es cuando todas las funcionalidades que permiten el descubrimiento de conocimiento en su totalidad se encuentran como una operación primitiva del SGDB posibilitándolo para desarrollar aplicaciones de este tipo [65].

1.3 IMPLEMENTACIÓN DE HERRAMIENTAS DCBD DÉBILMENTE ACOPLADAS CON SGBD

La implementación de herramientas DCBD débilmente acopladas con un SGBD se hace a través de órdenes SQL embebidas en el lenguaje anfitrión del motor de minería de datos [6]. Los datos residen en el SGBD y son leídos registro por registro a través de ODBC, JDBC o de una interfaz de cursores SQL. La ventaja de esta arquitectura es su portabilidad. Sus principales desventajas son la escalabilidad y el rendimiento. El problema de escalabilidad consiste en que las herramientas y aplicaciones bajo este tipo de arquitectura cargan todo el conjunto de datos en memoria, lo que las limita para el manejo de grandes cantidades de datos. El bajo rendimiento se debe a que los registros son copiados uno por uno del espacio de direccionamiento de la base de datos al espacio de direccionamiento de la aplicación de minería de datos [18] [31] que pueden llegar

a ser muy costosas computacionalmente figura 3, aunque con el avance en la velocidad y capacidad de los dispositivos computacionales cada vez más se han ido reduciendo los inconvenientes generados por estos factores.

Figura 3: Arquitectura DCBD débilmente acoplada.



2. CLUSTERING

Es una de las tareas de aprendizaje no supervisado en la que no se requiere una clasificación predefinida. El objetivo es particionar los datos obteniendo el conocimiento de acuerdo a las características de los mismos. En general, las clases de los datos no se presentan en el conjunto de datos y los objetos son agrupados basándose en el principio de maximización de similitud dentro de los clúster o grupos y minimización de similitud entre los diferentes clúster [32]. Esta técnica se ha venido estudiando hace 40 años al lado de muchas disciplinas, como las ciencias sociales, el aprendizaje de máquina y los métodos estadísticos, debido a su paso fundamental para generar una visión general de un conjunto de datos, dando la posibilidad de identificar distintas regiones y por lo tanto descubrir patrones y relaciones interesantes entre los atributos. Uno puede desear agrupar especies similares, sonidos, secuencias de genes, imágenes, señales, o registros de base de datos, entre las otras posibilidades. Sin embargo, Los mayores esfuerzos se han hecho con el fin de llevar a un agrupamiento eficaz y eficiente en grandes volúmenes de datos empezaron solamente hace unos cuantos años con el surgimiento de data mining [53].

En la minería de datos, el análisis de clústeres puede ser utilizado como una herramienta independiente para obtener una visión de la distribución de los datos, para observar las características de cada clúster y enfocar un análisis más exhaustivo hacia un grupo o clúster determinado [16]. Alternativamente, puede servir como un paso del preprocesamiento para otros algoritmos como por ejemplo para el de clasificación en el cual se trabajaría luego sobre los clústeres originados. El clustering de datos está en continuo desarrollo, y debido a los grandes volúmenes de datos almacenados, el análisis de clústeres se ha vuelto un tema altamente importante en los estudios de minería de datos [16].

2.1 CARACTERÍSTICAS DE LOS ALGORITMOS DE CLUSTERING

Las características deseables de la mayoría de los algoritmos de clustering son las siguientes [26]:

- ✓ **Escalabilidad:** La mayoría de los algoritmos de clustering trabajan de manera apropiada con un número pequeño de observaciones (hasta 200 aproximadamente), mientras que se necesita una gran escalabilidad para realizar agrupamiento de datos en bases con millones de observaciones.

- ✓ **Habilidad**: para trabajar con distintos tipos de atributos. Muchos algoritmos se han diseñado para trabajar sólo con datos numéricos, mientras que en una gran cantidad de ocasiones, es necesario trabajar con atributos asociados a tipos numéricos, binarios, discretos y alfanuméricos.

- ✓ **Descubrimiento de clúster con formas arbitrarias**: La mayoría de los algoritmos de clustering se basan en la distancia euclidiana, lo que tiende a encontrar clústeres todos con forma (circular) y densidad similares. Es importante diseñar algoritmos que puedan establecer clústeres de formas arbitrarias.

- ✓ **Requerimientos mínimos en el conocimiento del dominio para determinar los parámetros de entrada:** La herramienta no debería solicitarle al usuario que introduzca la cantidad de clases que quiere considerar, ya que dichos parámetros en muchas ocasiones no son fáciles de determinar, y esto haría que sea difícil controlar la calidad del algoritmo.

- ✓ **Habilidad para tratar con datos ruidosos:** La mayoría de las BD contienen datos con comportamiento extraño, datos faltantes, desconocidos o erróneos. Algunos algoritmos de clustering son sensibles a tales datos y pueden derivarlos a clústeres de baja calidad.

- ✓ **Insensibilidad al orden de las observaciones de entrada:** Algunos algoritmos son sensibles al orden en que se consideran las observaciones. Por ejemplo, para un mismo conjunto de datos, dependiendo del orden en que se analicen, los clústeres devueltos pueden ser diferentes. Es importante entonces que el algoritmo sea insensible al orden de los datos, y que el conjunto de clústeres devuelto sea siempre el mismo.

- ✓ **Alta dimensionalidad:** Una Base de Datos o un DataWarehouse puede contener varias dimensiones o atributos, por lo que es bueno que un algoritmo de clustering pueda trabajar de manera eficiente y correcta no sólo en repositorios con pocos atributos, sino también en repositorios con un alto espacio dimensional, o gran cantidad de atributos.

- ✓ **Clustering basado en restricciones:** Es un gran desafío el agrupar los datos teniendo en cuenta no sólo el comportamiento, sino también que satisfagan ciertas restricciones.

- ✓ **Interpretación y uso:** Los usuarios esperan que los resultados del clustering sean comprensibles, fáciles de interpretar y de utilizar.

Con estas características, se busca diseñar algoritmos más flexibles que sean capaces de manipular una gran variedad de requerimientos de acuerdo a las necesidades de los usuarios.

2.2 FORMAS DE MEDIR DISTANCIA EN ESPACIOS MULTIDIMENSIONALES

Ya que el objetivo del conglomerado es agrupar objetos similares, se necesita alguna medida para evaluar las diferencias y similitudes entre objetos. La estrategia más común consiste en medir la equivalencia en términos de la distancia entre los pares de objetos. Los objetos con distancias reducidas entre ellos son más parecidos entre sí, que aquellos que tienen distancias mayores. Existen varias formas de calcular las distancias entre dos objetos, que dependen del tipo de variables, ya sean numéricas o cuantitativas cuando el valor de una variable se expresa numéricamente como la edad de una persona, el peso y categóricas o cualitativas cuando su valor representa una cualidad o una característica como el color de ojos, el género y los valores están divididos en categorías. Hay que tener en cuenta que para las variables numéricas se utilizan funciones de distancia debido a sus características geométricas y para las variables categóricas se utilizan medidas de similitud o también de distancia entendiendo distancia como las diferencias que hay entre dos objetos[89].

2.2.1 Medidas de distancia para variables numéricas

La distancia entre dos elementos de un conjunto de datos $\mathbb{R}$ se define como cualquier función binaria $d(a, b)$ de $\mathrm{X} \times \mathrm{X}$ en $\mathbb{R}$, que verifique las siguientes condiciones [57] [101]:

- No negatividad, el resultado no puede ser un valor negativo:

$$d(a,b) \geq 0 \ \forall a, b \in X \quad (1)$$

- Simétrica, El resultado debe ser simétrico en sus argumentos :

$$d(a,b) = d(b,a) \ \forall a, b \in X \quad (2)$$

- Desigualdad triangular, el resultado debe verificar que si se tiene tres puntos, la suma de las longitudes de dos lados cualesquiera del triangulo formado por los tres puntos debe siempre ser mayor que el tercer lado.

$$d(a,b) \leq d(a,c) + d(c,b) \ \forall a, b, c \in X . \quad (3)$$

- El resultado entre un elemento y el mismo es cero.

$$\forall x \in X : d(x,x) = 0$$

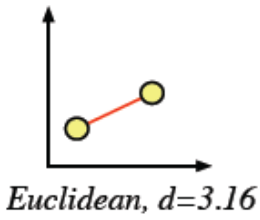
Si $x, y \in X$ son tales que $d(x, y) = 0$, entonces $x = y$. (4)

Distancia Euclidiana

La distancia euclidiana es la longitud del camino más corto entre dos puntos. Es decir, la medición del grado de cercanía que existe entre los dos. La distancia euclidiana se define como la raíz cuadrada de la suma de las diferencias cuadradas de los valores para cada variable. Esta distancia se deduce a partir del teorema de Pitágoras y que se puede ver en la figura 4 [28] [74] [102].

$$d(i,j) = \sqrt{|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \ldots + |x_{in} - x_{jn}|^2} \quad (5)$$

Figura 4: Representación grafica de la distancia Euclidiana.



Distancia Manhattan

La distancia Manhattan o City Block es la distancia entre dos puntos medida a lo largo de los ejes en ángulo recto (ver figura 5). Se define como la suma de las diferencias absolutas de los valores para cada variable de dos objetos [28] [85].

$$d(i,j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \ldots + |x_{in} - x_{jn}| \quad (6)$$

Figura 5: Representación grafica de la distancia Manhattan.

Distancia Chebyshev

Se define como la mayor diferencia absoluta que puede existir entre los las variables de dos objetos [7].

$$D(i,j) = \text{Max}|x_{ki} - x_{kj}| \quad (7)$$

Distancia Mahalanobis

Es una medida de distancia introducida por Mahalanobis en 1936. Su utilidad radica en que es una forma de determinar la similitud entre dos variables multidimensionales. Se diferencia de la distancia euclidiana en que tiene en cuenta la varianza (ver formula 8) o la covarianza (ver formula 9) que hay entre las variables por esta razón también es conocida como la distancia euclidiana estandarizada. También puede servir para establecer la relación entre las variables de los conjuntos de datos combinando sus varianzas o covarianzas[104].

Utilizando en la distancia mahalanobis la varianza quedaría expresada de la siguiente forma en expresión vectorial:

$$d_e(\vec{x_1}\vec{x_2}) = \sqrt{(\vec{x}_1 - \vec{x}_2)^T S^{-1}(\vec{x}_1 - \vec{x}_2)} \quad (8)$$

Utilizando en la distancia mahalanobis la covarianza quedaría expresada de la siguiente forma en expresión vectorial:

$$d_m(\vec{x_1}\vec{x_2}) = \sqrt{(\vec{x}_1 - \vec{x}_2)^T \Sigma^{-1}(\vec{x}_1 - \vec{x}_2)} \quad (9)$$

2.2.2 Medidas de distancia o similitud para variables categóricas

Distancia Hamming

Se denomina distancia Hamming o distancia de similitud, a la semejanza de los valores de un vector con otro y su resultado depende de la diferencias entre los dos elementos, si son iguales o no. Cuanto mayor sea esta diferencia, menor es la posibilidad de que sean similares debido a una serie de diferencias que se pueden definir como errores. Esta función de similitud se define como el número de

valores que tienen que cambiarse para transformar de un vector para ser igual a otro. Si dos vectores tienen una diferencia de 2, se dice que para ser iguales necesita cambiar dos valores. Para el caso de los conjuntos de datos se toman como vectores u objetos los registros del conjunto de datos y se compara atributo por atributo, si en un atributo coinciden con el valor los dos registros se suma un 0 y si son diferentes se suma un 1(ver formula 11). Es decir si un objeto es igual a otro la distancia será 0, pero si no es así el valor resultante será el numero de atributos en el cual difieren [104] [66].

Figura 6: Función de similitud.

$$d(X_1, X_2) = \sum_{j=1}^{p} \delta(x_{1,j}, x_{2,j}), \tag{10}$$

Donde:

$$\delta(x_{1,j}, x_{2,j}) = \begin{cases} 0 & \text{if } x_{1,j} = x_{2,j}, \\ 1 & \text{if } x_{1,j} \neq x_{2,j}. \end{cases} \tag{11}$$

Proporción de coincidencias

Se calcula como el número total de coincidencias sobre el número total de variables, donde m es el número de valores iguales en las variables del objeto i y j y p es el número total de variables. También se puede encontrar el grado de disimilitud, encontrando la diferencia entre el número total de variables menos el numero de coincidencias sobre el número total de variables [57][28].

$$d(i, j) = \frac{p - m}{p} \tag{12}$$

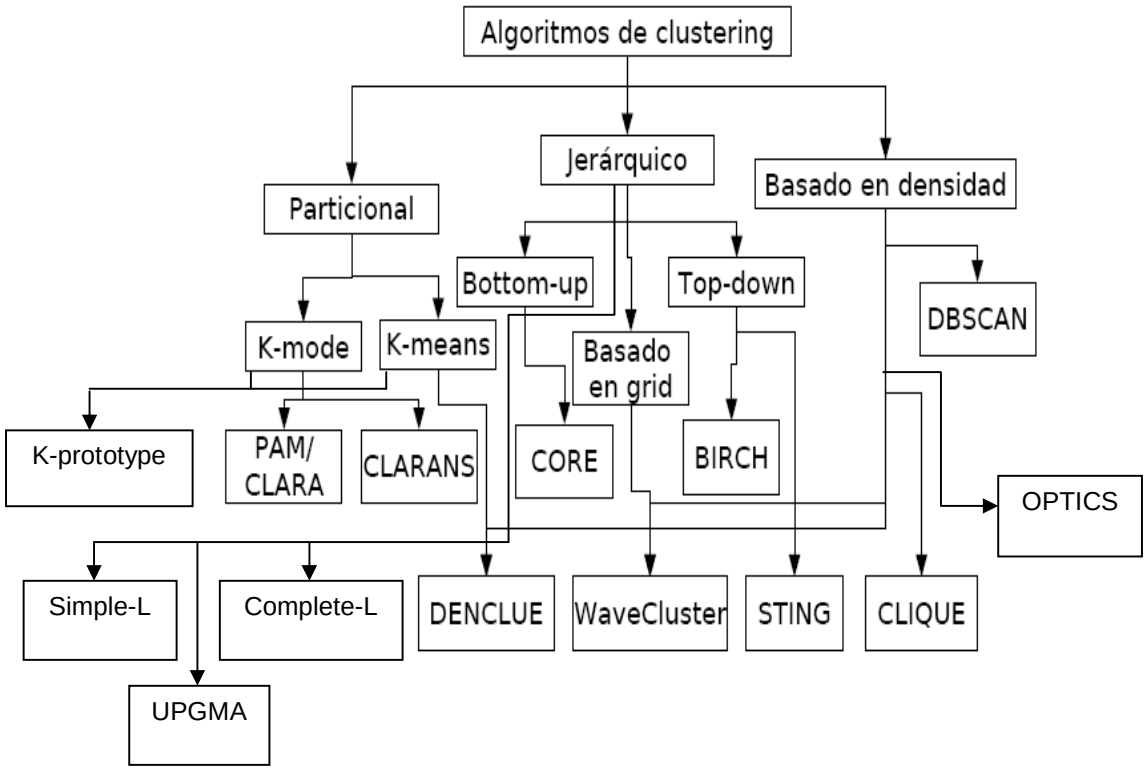
2.3 TÉCNICAS O MÉTODOS DE CLUSTERING

Las técnicas de Clustering varían dependiendo del tipo de datos con el cual se va a trabajar, es decir, es importante tener en cuenta que tipo de información se manejara, si son datasets, sonidos, imágenes…, también hay que tener en cuenta de que tipo de datos están compuestos, es decir si son un tipo de datos numérico, ya sea continuo o discontinuo, o bien pueda ser de tipo categórico o cualitativo, es

decir atributos como el color, estado civil, tipo de sangre…, que podrían ser dicotómicos o politomicos. Teniendo en cuenta la variedad de fuentes datos, los diferentes tipos de datos, se han ido desarrollando técnicas que varían entre sí por las reglas heurísticas para lograr la proximidad entre objetos. La mayoría de ellos se basa en el empleo sistemático de distancias entre vectores (objetos a agrupar) así como entre clúster o grupos que se van formando a lo largo del proceso de clustering [33] [53].

En la literatura existen una gran cantidad de técnicas de clustering que varían de acuerdo a la arquitectura que utiliza [43]. Una clasificación general divide los algoritmos en: clustering particional, clustering jerárquico, clustering basado en densidad y clustering basado en grillas y demás (ver figura 7). [52][33].

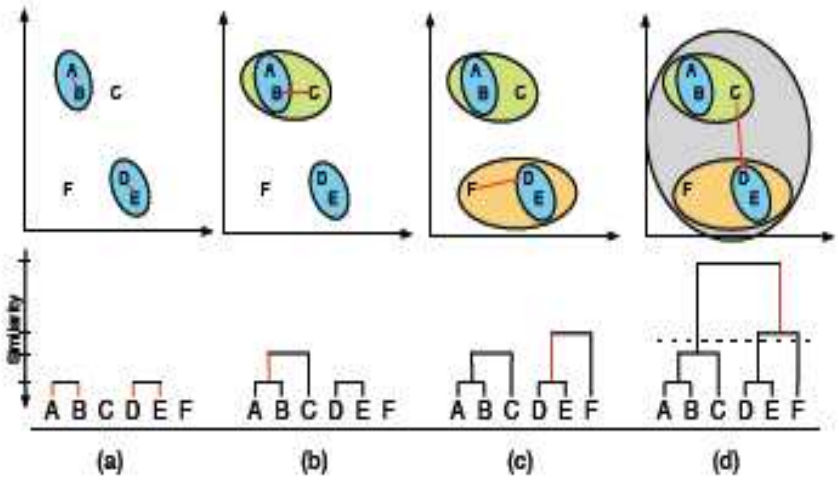
Figura 7: Clasificación de algoritmos básicos de Clustering.



2.3.1 Clustering jerárquico

Un método jerárquico crea una descomposición jerárquica de un conjunto de datos, formando un dendograma (árbol) que divide recursivamente el conjunto de datos en conjuntos cada vez más pequeños o más grandes [43]. La figura 8 muestra la representación gráfica de un dendograma y forma como se realizan las uniones de los grupos.

Figura 8: Dendograma de un conjunto de datos.



En la parte inferior del dendograma se disponen los n elementos iníciales, las uniones entre los elementos se indican por tres líneas rectas. Dos dirigidas a los elementos que se unen, y que están de forma perpendicular al eje de los elementos y una paralela a este eje, que sitúa el nivel en que se unen. Las agrupaciones resultantes son determinadas por el corte del dendrograma en un cierto nivel.

El árbol puede ser creado de dos formas: de abajo hacia arriba (bottom-up) o de arriba hacia abajo (top-down) (ver figura 9). En el caso bottom-up, también llamado aglomerativo, se comienza con cada objeto formando un grupo por separado. Los objetos o grupos se combinan sucesivamente según determinadas medidas, hasta que todos los grupos se hayan unido en uno solo, o hasta que se cumpla alguna condición de terminación. En el caso top-down, también llamado divisivo, se comienza con todos los objetos en el mismo clúster, y a medida que se va iterando, se dividen los clústeres en subconjuntos más pequeños según determinadas medidas, hasta que cada objeto se encuentre en un clúster individual o hasta que se cumplan las condiciones de terminación.

Los algoritmos aglomerativos requieren menos tiempo de cálculo y son más utilizados [57]. Los algoritmos aglomerativos realizan la unión de los clúster teniendo en cuenta las distancias entre clústeres, que pueden ser Single Link, Average Link y Complete Link que son las formas básicas (ver figura 10) [11]. [55l][51].

Figura 9: Formas crear el Dendograma.

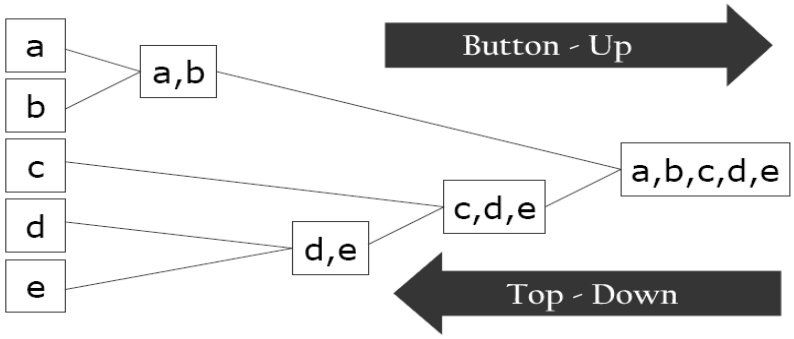
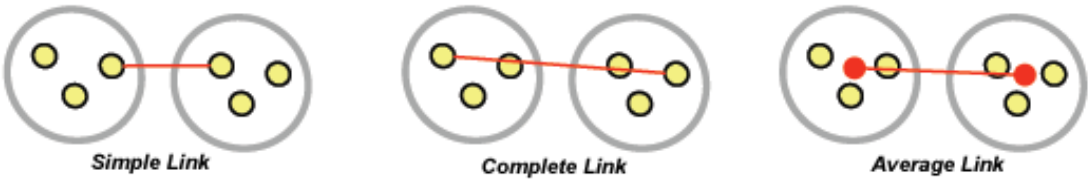


Figura 10: Definiciones de distancia entre grupos.



Single Link

En cada paso se unen los dos grupos cuyos elementos más cercanos tienen la mínima distancia.

Average Link

En cada paso se unen los dos grupos tal que tienen la mínima distancia promedio entre sus puntos.

Complete Link

En cada paso se unen los dos grupos tal que su unión tiene el diámetro mínimo o los dos grupos con la menor distancia máxima entre sus elementos.

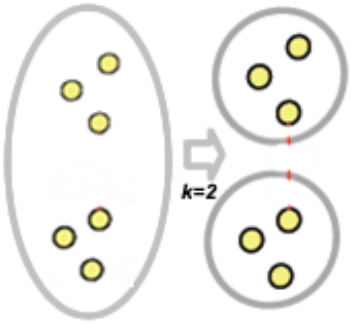
Otras formas de realizar uniones entre grupos son el método del centroide, que consiste en calcular la distancia entre dos grupos mediante una función de distancia entre sus centros, en donde se toman como centros los vectores de medias o modas de los elementos que forman el clúster. También se puede encontrar un método denominado Ward el cual es un proceso diferente de construir clústeres, suponiendo la unión de grupos y buscando el mínimo incremento de una función Ward.

Algunos algoritmos de clustering que pertenecen a esta clasificación son: CURE (Clustering Using Representatives) [29], CHAMALEON [45], BIRCH (Balanced Iterative Reducing and Clustering using Hierarchical) [68] y ROCK (RObust Clustering algorithm using linKs) [30].

2.3.2 Clustering particional

Un algoritmo de clustering particional obtiene una partición simple de los datos en k grupos [43]. Los métodos particionales aparecieron a comienzos de la historia del clustering, mucho antes de la aparición de la minería de datos. En la figura 11 se muestra un ejemplo de clustering particional con k igual a dos.

Figura 11: Clustering particional.



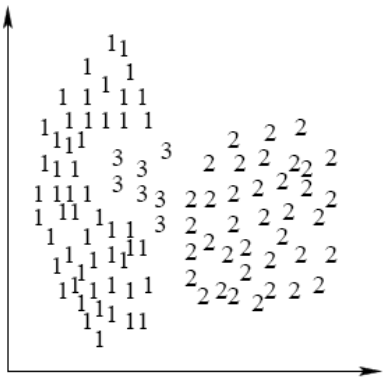
Se dice que es un método de agrupación de clústeres duros. Esto significa que un punto de datos puede pertenecer a un solo clúster, y que únicamente se calcula una probabilidad de pertenencia de cada punto de datos de ese clúster. El clustering particional organiza los objetos dentro de k clúster de tal forma que sea

minimizada la desviación total de cada objeto desde el centro de su clúster o desde una distribución de clústeres. La desviación de un punto puede ser evaluada en forma diferente según el algoritmo, y es llamada generalmente función de similitud o función objetivo. Los métodos particionales tienen ventajas en aplicaciones que involucran gran cantidad de datos para los cuales la construcción de un dendograma resultaría complicada El problema que se presenta al utilizar algoritmos particionales es la decisión del número deseado de clústeres de salida. Las técnicas particionales usualmente producen clústeres que optimizan el criterio de función definido local o globalmente. En la práctica, el algoritmo se ejecuta múltiples veces con diferentes estados de inicio y la mejor configuración que se obtenga es la que se utiliza como el clustering de salida. Algunos algoritmos de clustering que pertenecen a esta clasificación son: CLARA (Clustering Large Applications) [46], CLARANS (Clustering Large Applications based on Randomized Search) [27], K-prototype [50], K-mode [37], K-Means [39].

2.3.3 Clustering basado en densidad

Los algoritmos basados en densidad obtienen clústeres basados en regiones densas de objetos en el espacio de datos que están separados por regiones de baja densidad (estos elementos aislados representan ruido), aquí “densidad” se refiere al concepto físico de densidad en lugar de una distribución estadística particular [49]. En la figura 12 se muestra un ejemplo de clustering basado en densidad.

Figura 12: Clustering basado en densidad.



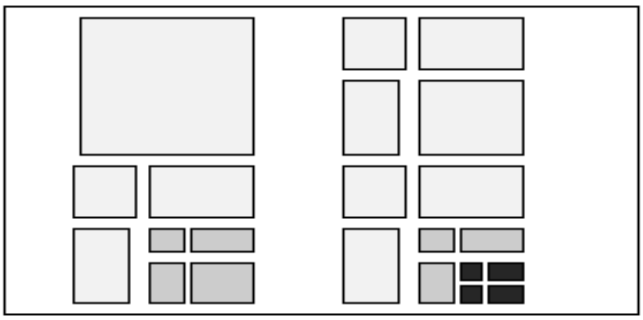
Este tipo de algoritmos comienzan la búsqueda de objetos básicos, y son cada vez más los objetos que se forman sobre la base de estos núcleos y siguen tras la búsqueda de objetos que se encuentran en un vecindario dentro de un radio épsilon de un determinado objeto. La ventaja de este tipo de algoritmos es que puede detectar de forma arbitraria agrupaciones y puede filtrar el ruido [11],

además los métodos basados en densidad pueden descubrir las agrupaciones con formas arbitrarias y no es necesario dar un número preestablecido de agrupaciones. Algunos algoritmos de clustering que pertenecen a esta clasificación son: DBSCAN (Density-Based Spatial Clustering of Aplications with Noise) [22], OPTICS (Ordering Points to Identify the Clustering Structure) [9] y DENCLUE (DENsity-based CLUstEring) [36].

2.3.4 Clustering basado en grid o grillas

Recientemente un número de algoritmos de clustering han sido presentados para datos espaciales, éstos son conocidos como algoritmos basados en grid [43]. Estos algoritmos cuantifican el espacio en un número finito de celdas y aplican operaciones sobre dicho espacio. La mayor ventaja de este método es su veloz procesamiento del tiempo, el cual generalmente es independiente de la cantidad de objetos a procesar. En la figura 13 se muestra un ejemplo de clustering basado en grid.

Figura 13: Clustering basado en grid.

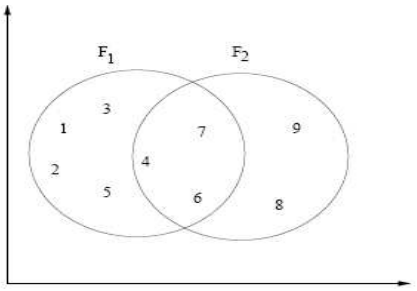


Algunos algoritmos de clustering que pertenecen a esta clasificación son: STING (Statistical Information Grid-based method) [66], CLIQUE [1] y Waveclúster [61].

2.3.5 Clustering difuso

En los algoritmos de clustering tradicionales, cada patrón pertenece única y exclusivamente a un solo clúster. El clustering fuzzy asocia cada patrón con cada clúster utilizando funciones de pertenencia [67]. La salida de cada algoritmo es un agrupamiento, no una partición excluyente. La figura 14 muestra un ejemplo de clustering difuso generando dos clústeres F1 y F2 cuyos datos tienen un nivel de pertenencia en los 2 clústeres.

Figura 14: Clustering difuso.



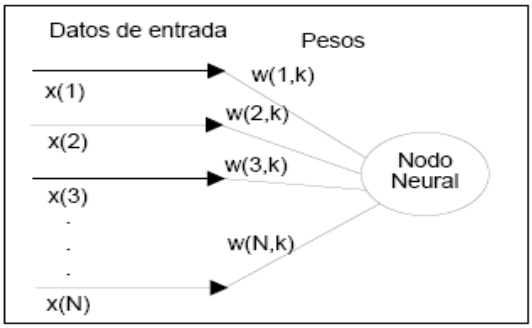
2.3.6 Clustering de redes neuronales artificiales

Las redes neuronales artificiales (ANN) [35], fueron motivadas por las redes neuronales biológicas. Las ANN tienen una extensa utilización tanto en técnicas de clasificación como de clustering en minería de datos. Dentro del proceso de clustering las ANN's presentan las siguientes características:

- ✓ Procesan vectores numéricos y por lo tanto requieren patrones para poder representarse utilizando únicamente características cuantitativas [43].
- ✓ Son inherentemente paralelas y utilizan arquitecturas de procesamiento distribuido.
- ✓ Pueden aprender por medio de la interconexión y adaptabilidad de los pesos.
- ✓ Más específicamente, pueden actuar como patrones normalizados y selectores si se seleccionan adecuadamente los pesos.

La arquitectura de estas redes es simple: todas tienen una sola capa. En la figura 15 se muestran las conexiones de los pesos de entrada en un nodo de la red.

Figura 15: Conexión de los datos de entrada a un nodo neural.



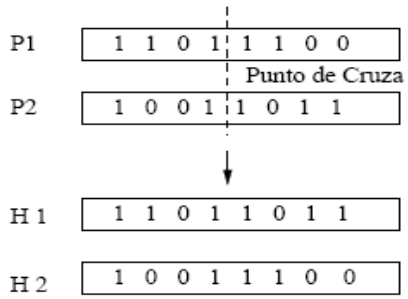
Los patrones se introducen en la entrada y son asociadas a los nodos de salida. Los pesos entre los nodos de entrada son iterativamente modificados hasta que se satisfaga el criterio de terminación. El aprendizaje competitivo tiene su semejanza en las redes neuronales biológicas.

Las ANN más utilizadas en clustering son los vectores de cuantización de aprendizaje de Kohonen (learning vector quantization -LVQ), los mapas de auto-organización de Kohonen (self-organizing map -SOM) [47][34] y modelos adaptativos de resonancia (adaptive resonance theory - ART) [17].

2.3.7 Clustering de algoritmos evolutivos.

Los algoritmos evolutivos, motivados por la evolución natural, utilizan operadores genéticos y una población de soluciones para obtener el óptimo global en la partición de los datos [43]. Los operadores genéticos más utilizados son: selección, cruza y mutación. La figura 16 muestra la operación de cruza del proceso evolutivo entre los padres P1 y P2 generando los hijos H1 y H2.

Figura 16: Operación de cruza de un modelo evolutivo.



2.4 VALIDEZ DE LOS CLÚSTERES.

Las técnicas de clustering a diferencia de otras técnicas como la clasificación supervisada en las que hay medidas de calidad del modelo desarrollado, como lo es la probabilidad de error, en el análisis de clústeres, la pregunta análoga es cómo evaluar la “bondad” o la calidad de los clústeres que resultan. La diferencia radica en que el clustering puede ser utilizado para evaluar diferentes aspectos como:

- ✓ Buscar patrones en el ruido
- ✓ Comparar algoritmos de clustering
- ✓ Comparar dos conjuntos de clústeres
- ✓ Comparar formaciones de clústeres

Para algunos de estos aspectos, se puede evaluar tanto a nivel del clustering general con todas sus formaciones o al nivel de un clúster particular. Lo ideal es que cuando se apliquen varias técnicas de clustering se pueda ir encontrando formaciones estables, es decir que no varían en su conformación, que se mantienen estables. Y se pueda ir identificando datos atípicos. Hay que tener en cuenta al evaluar los resultados de las agrupaciones que no se puede utilizar un mismo concepto para los atributos de tipo numérico, que para los atributos categóricos.

Para los atributos numéricos una forma de cumplir con los principios del clustering es el análisis del error cuadrático [57][8], que es la diferencia entre los miembros del clúster y la media, la suma de todas estas diferencias determina que tan unidos están los objetos de ese clúster. Esta fórmula (12) es conocida como la formula de Ward. Se dice que un clúster es mejor a medida que el resultado de esta sumatoria sea menor. El valor óptimo de agrupación es cero.

$$WSS = \sum_{i} \sum_{x \in C_i} (x_i - m_i)^2 \quad (13)$$

Donde la x_i es elemento de clúster C_i y m_i es la media del clúster C en el atributo i.

Para los atributos de tipo categórico hay que evaluar la cantidad de desajustes que se tengan en el valor de la moda [20], es el valor restante para que la frecuencia (ver formula 14) de la moda sea 1. Entre más se acerque este valor a cero, mejor va ser la agrupación.

$$\sum_{k=1}^{k} \sum_{j=1}^{m} (1 - f_{kj}) \quad (14)$$

Donde f_{kj} es la frecuencia de la moda en el atributo j del clúster k.

Hay que tener en cuenta que los valores de estos índices dependen mucho a los atributos que se va aplicar sobre todo en los atributos de datos numéricos, ya que al trabajar a diferentes escalas algunos atributos pueden influir directamente al resultado. Sobre todo en los conjuntos de datos mixtos.

Uno de las frases más reconocidas en el clustering es "*La validación de las estructuras de agrupación es el tarea más difícil y frustrante de la parte de análisis de clustering. Sin un gran esfuerzo en esta dirección, el análisis de clústeres seguirá siendo un arte negro accesible sólo a los creyentes que tienen experiencia y una gran valentía*" dicha por Jian en 1988 [44], que resume lo complejo del análisis de los modelos de clustering.

2.5 APLICACIÓNES DE LOS ALGORITMOS DE CLUTERING

La búsqueda automática de los grupos naturales en los datos, es una técnica de exploración de los datos muy importante. Tiene muchas aplicaciones en diferentes áreas de las ciencias naturales y sociales. Sin embargo, los esfuerzos para llevar a cabo eficaces y eficientes mecanismos de clustering en grandes conjuntos de datos sólo comenzaron en los últimos años con el surgimiento de la minería de datos [53]. En los proyectos de minería de datos, debe ser una de las primeras tareas a realizar, con el fin de ver las tendencias de los datos y ver qué tipo de registros o atributos se encuentran más relacionados. Las técnicas de clustering pueden ser instrumentos útiles para explorar la estructura subyacente de un determinado conjunto de datos y se aplican en una amplia variedad de disciplinas. Se puede argumentar que no habrá un algoritmo de agrupación universal que podría aplicarse a datos arbitrarios. Por lo tanto, es importante tener una clara comprensión de las necesidades de una aplicación y elegir la técnica que mejor se adapte a esos requisitos. En general, para ser útiles en aplicación, un algoritmo de agrupamiento debe tener las siguientes habilidades [1]:

- ✓ Según los tipos de datos a agrupar (numérico, categórico, texto e imágenes).
- ✓ Calidad de los datos a manejar, es decir el ruido que contengan, los datos anómalos y difusos.
- ✓ Cantidad de datos a trabajar, sus dimensiones
- ✓ Tener muy en cuenta que resultado se espera.
- ✓ ser fácil de aplicar.

Como se ha mencionado anteriormente el clustering ha incursionado en varias disciplinas siendo aplicado para la recuperación de información, clasificación de documentos, creación de catálogos, tratamiento de imágenes, aplicaciones en nuevos paradigmas en los buscadores web, segmentación de mercados y demás. A continuación se presentan algunas aplicaciones de las técnicas de clustering particional, clustering jerárquico y clustering basado en densidad en donde sus condiciones pueden presentar un mejor resultado.

2.5.1 Aplicación de técnicas de clustering particional

Este tipo de técnicas son ideales cuando se cuenta con un conjunto de datos de gran tamaño proveniente de diferentes fuentes, gracias a su buen rendimiento, ya que se pueden crear subconjuntos de datos con características similares, con el fin de aplicar a estos subconjuntos algoritmos de clasificación o asociación y poder obtener mejores reglas o patrones. Como por ejemplo en la investigación ”Aplicación De Minería De Datos Para La Exploración Y Detección De Patrones Delictivos En Argentina”[58] realizada en el Instituto Tecnológico de Buenos Aires, en Argentina, se utilizo el clustering aplicándolo en el conjunto de datos de información criminal de Argentina. Conjunto de datos que contaba con 1810 registros y 8 atributos, aquí se utilizo el algoritmo kmeans para generar 3 clústeres, teniendo como resultado los valores que se pude ver en la figura 17. En esta investigación se concluyo que hay cierta alternancia entre las modas de los atributos *lugar, arma* y *otro delito* que parecerían estar identificando a los clústeres. Enseguida se aplico el algoritmo de clasificación C4.5 teniendo como atributo clase el atributo clúster y variando los atributos evaluados, obteniendo como resultado que se tiene un “excelente poder clasificatorio de esta combinación (98,8% de instancias bien clasificadas) confirma que la los *clústeres* determinados por *K-means* responden a un criterio determinado subyacente a los datos”. Comprobando así la importancia de utilizar las técnicas de clustering, como etapa inicial de un proyecto de minería de datos.

Figura 17: Tabla de centroides, después de ejecutar el algoritmo kmeans de la herramienta Weka en el conjunto de datos de información criminal de Argentina.

		Atributos categóricos (modas)				Atributos continuos (medias)			
	Cant. (%)	Provincia	Lugar	Arma	Otro Delito	Hora	Día Semana	Día Mes	Mes
Cluster 0	22%	BsAs	Vía Pública	de Fuego	Robo	19	Sábado	16	7
Cluster 1	43%	BsAs	Vía Pública	de Fuego	No Hubo	17	Sábado	15	7
Cluster 2	35%	BsAs	Domicilio Particular	Blanca	No Hubo	21	Sábado	15	7
General	100%	BsAs	Vía Pública	de Fuego	No Hubo	19	Sábado	15	7

2.5.2 Aplicación de técnicas de clustering jerárquico

Esta técnica de clustering permite crear de forma automática clasificaciones de objetos considerando ciertas similitudes en sus características. Esta técnica es muy útil cuando se quiere hacer una taxonomía de los datos, o determinar un cierto orden en los datos, un ejemplo de ello es la investigación denominada "FARC Terrorism in Colombia: A Clustering Analysis" [10] realizada por el Ministerio de hacienda de la república de Colombia, en el cual se aplico clustering jerárquico con enlace-máximo a la base de datos obtenida en el Ministerio de Defensa y el Ministerio del Interior y Justicia de Colombia. La cual contiene información de los actos terroristas, especialmente cometidos por las FARC entre enero 2 de 1999 y 9 de febrero de 2003. La base de datos contiene 5.819 actos que se pueden dividir en 117 tipos diferentes (por ejemplo: secuestros, masacres, atentados a conductos de petróleo, atentados a torres de energía, los ataques contra la infraestructura militar, etc.), con el fin de presentar información para el apoyo de tomas de decisiones no sólo para las futuras estrategias militares, sino también para determinar una mejor priorización y asignación de los escasos recursos militares.

En el caso de bases de datos documentales como los catálogos de bibliotecas [15], se busca agrupar dentro de un mismo *clúster* aquellos documentos en los que coocurran mayor número de términos de indización, por ejemplo. Si a estas técnicas de agrupación se suma una presentación gráfica de su resultado, se obtiene una nueva forma de ver los documentos de las bases de datos que facilita el ojeo y por tanto la recuperación de información y esto se logra con el dendograma. Esta técnica permite al usuario moverse por la base de datos en función de sus intereses y sin necesidad de definir previamente su objeto de búsqueda como ocurre en las búsquedas convencionales. Scatter/Gather y Clustifier son dos ejemplos de sistemas que emplean el *clustering* como técnica de presentación de resultados, si bien no incorporan una visualización gráfica. Algunos OPACs (inglés Online public access catalog o catálogo automatizado de acceso público en línea de los materiales de una biblioteca) y motores de búsqueda han hecho grandes avances en esta línea como Vivísimo, que es un buscador creado el año 2000 que, a partir de una tecnología propia (Advanced document clustering) es capaz de dividir por categorías los resultados de la búsqueda. La tilde en la segunda i de Vivísimo es un gesto de inequívoca reafirmación hispana de este potente buscador, que clasifica la información encontrada mediante una combinación de análisis lingüísticos y estadísticos.

Vivísimo ofrece a las empresas soluciones de búsqueda que permiten a los usuarios finales encontrar y aprovechar toda la información disponible, independientemente de su origen, la ubicación o tipo. Esta empresa ha creado un metabuscador llamado clusty que han visto en el *clustering* una técnica de presentación de los resultados muy beneficiosa para mostrar los documentos en

categorías según la cantidad de términos que coinciden en sus textos. En la figura 18 se presenta la página que se presenta al buscar en http://clusty.com/ la palabra “Clustering” [76]. En la parte superior podemos encontrar los modos de búsqueda ya sea la web, news, wikipedia, blog, jobs o more según lo que el usuario desea encontrar, en la casilla esta la palabra a buscar.

Figura 18: Imagen de la página http://clusty.com/ con la búsqueda de la palabra “Clustering”

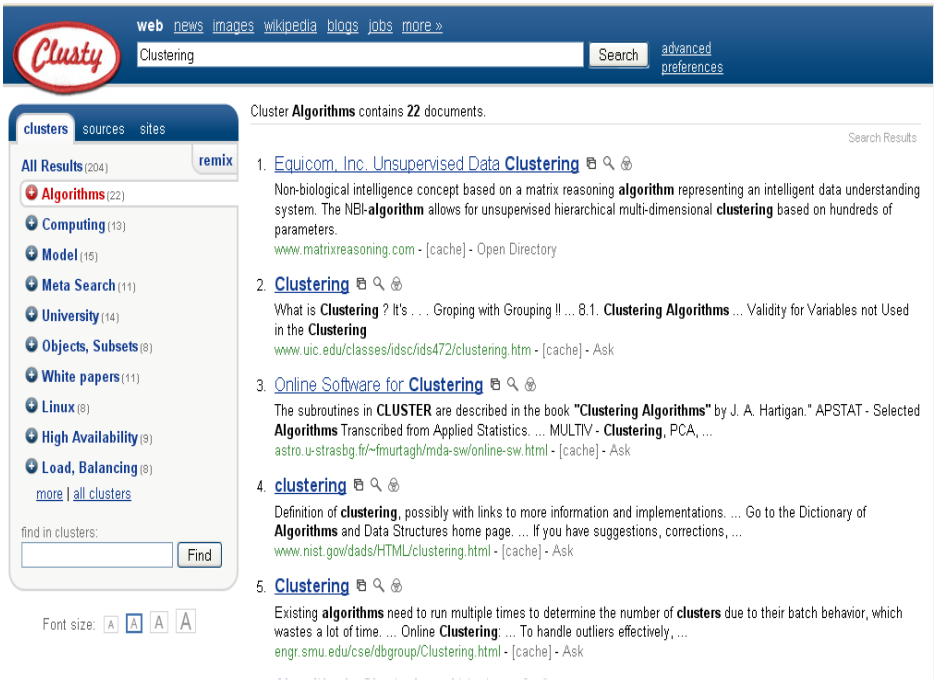
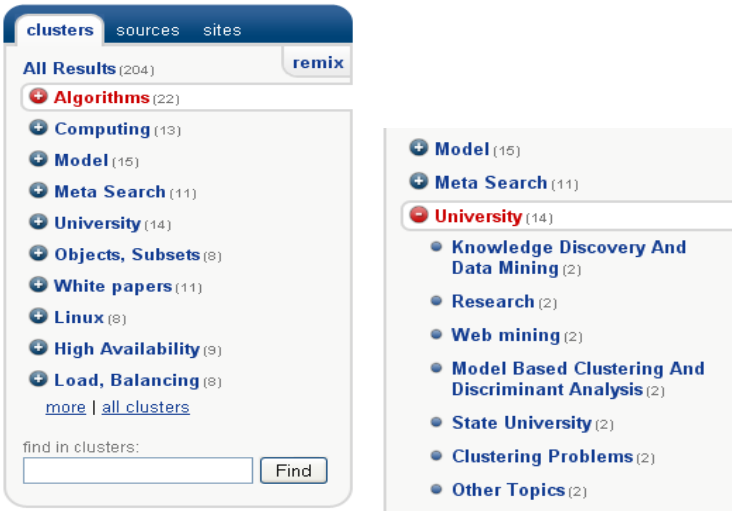


Figura 19: Clústeres en donde se encuentra la palabra “Clustering”.

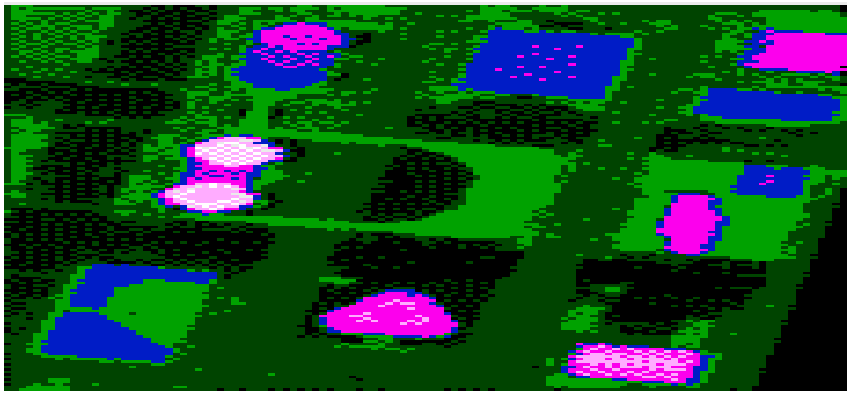


Uno de los buscadores más reconocido en el mundo es google, que utiliza este tipo de clustering y no solo esta técnica sino una combinación de varias de ellas, agrupando su información en periodos de tiempo considerables categorizándola para buscar la forma de presentarle al usuario mejores resultados en sus consultas [81]. El clustering jerárquico también es bien utilizado en las áreas que tengan que ver con biología y química [84], ya que permite generar taxonomías que muy difícilmente se lograrían de forma manual teniendo en cuenta todas las características que ellos tienen. Un ejemplo de ello es en Transcriptómica, que utiliza la agrupación para crear grupos de genes relacionados con patrones de expresión (también conocidos como genes coexpressed). A menudo estos grupos contienen proteínas funcionalmente relacionadas, como las enzimas para una cadena, o los genes que son co-regulados. Experimentos de alto rendimiento utilizando las etiquetas de secuencias expresadas (EST) o microarrays de ADN en donde pueden ser una herramienta poderosa para la anotación del genoma, un aspecto general de la genómica.

2.5.3 Aplicación de técnicas de clustering basados en densidad

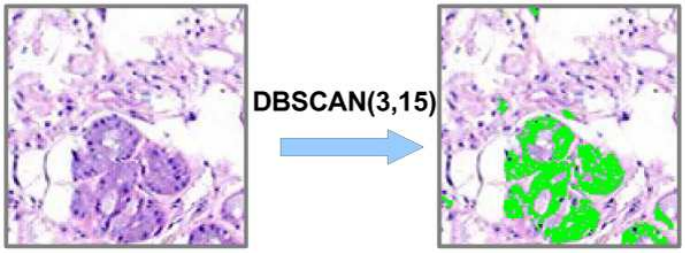
Aquí el algoritmo más utilizado es el DBSCAN. Para la aplicación de esta técnica se debe conocer muy bien el conjunto de datos que se desee trabajar y el resultado que se desee lograr. Ya que parten del principio de encontrar zonas con un alto grado de densidad de objetos. También es muy útil cuando el conjunto de datos contenga varios objetos anómalos o con ruido y también en las bases de datos espaciales. Es muy útil en el tratamiento de imágenes un ejemplo de ello es la creación de modelos de espacio geográfico (Espacio Modelos) para ser utilizado en el contexto sistemas móviles. Aquí, el papel de los algoritmos de agrupamiento es identificar grupos de puntos y a continuación utilizar los clústeres para automáticamente caracterizar las regiones geográficas. Ver figura 20[22] [80].

Figura 20: Imagen tomada desde el aire, reconociendo objetos con algoritmos de clustering.



Otra aplicación es en el análisis de imágenes histológicas para identificar posibles cuadros cancerígenos, creando nuevos métodos automático para hacer frente a carcinoma de células escamosas. La imagen que está en el lado izquierdo es la imagen toma por el microscopio y la imagen que está al lado derecho están las áreas o clústeres identificados con el algoritmo DBSCAN que presumen una anomalía [84].

Figura 21: Imagen histológicas tomadas desde un microscopio computarizado, mostrando la identificación de áreas anómalas a través del algoritmo DBSCAN.



Otra aplicación de las tareas de clustering basadas en densidad es en los sistemas Peer-to-Peer más conocidos como p2p, que sirven para compartir archivos de todo tipo. Estos sistemas almacenan las características de sus usuarios dependiendo de sus búsquedas realizadas, se van creando clústeres con usuarios con características similares y geográficamente cercanas, agilizando los procesos de descarga, ya que se busca primero en los puntos con características similares además de la ubicación geográfica, esta combinación de técnicas es aplicada en el algoritmo PENS (Peer dENsity – based cluStering) [48].

3. ALGORITMOS DE CLUSTERING IMPLEMENTADOS EN RASEMUS

Rasemus es una herramienta de clustering que aborda varias técnicas de clustering como la técnica de partición, jerárquica y basada en densidad, brindando al usuario la posibilidad de tener varias formas de clusterización, permitiéndole al usuario evaluar varios modelos descriptivos y diferentes perspectivas para analizar conjuntos de datos. El usuario puede elegir un algoritmo a utilizar dependiendo los tipos de datos a trabajar, la cantidad de instancias u objetos a trabajar y el tipo de pretensiones que desee alcanzar. Además se presentan dos propuestas de algoritmos de formación de clustering denominadas wayCluster y K-Frequency desarrolladas en laboratorio GRIAS de la línea KDD para la formación de clústeres de tipo particional, que se encuentran en proceso de evaluación.

Los algoritmos de clustering generalmente para la formación clústeres utilizan formulas de distancia o similitud dependiendo del tipo de datos a trabajar según la fórmula a utilizar puede variar el resultado y la calidad del mismo, por este motivo Rasemus implementa diferentes modos de medición teniendo en cuenta el tipo de variable a trabajar, si es de tipo numérico es usuario puede utilizar las formulas de distancia euclidiana, distancia mahattan, distancia de chebyshev y la distancia mahalanobis, para los atributos de tipo categórico se utilizan las funciones de similitud conocidas como distancia hamming y la proporción de coincidencias, que se mencionan en capitulo 2.2. Además si el usuario posee un conjunto de datos mixto, los algoritmos calculan la distancia entre atributos de tipos de datos numéricos y calcula la similitud entre atributos de tipo de datos categóricos, y al final hacen la suma de las dos funciones para encontrar la distancia total, por ejemplo la formula 13 presenta la combinación entre la distancia euclidiana y una formula de similitud teniendo que, la distancia entre los objetos x y y es la distancia que se encuentra en la diferencia al cuadrado de los pn atributos de tipo numérico, más la disimilitud δ de los atributos categoricos, donde m es el numero total de atributos. Esta forma de tratar los tipos de datos mixtos hace parte del algoritmo K-Prototype que se tratara posteriormente, pero por su manera práctica de calcular los distancia entre objetos con atributos mixtos se extendió a los demás algoritmos.

$$d(x,y) = \sum_{j=0}^{pn} (x_j - y_j)^2 + \sum_{j=pn+1}^{m} \delta(x_j, y_j) \quad (13)$$

Rasemus implementa las técnicas de clustering más utilizadas e implementa los algoritmos que contengan la filosofía básica de cada una de ellas, además de presentar dos propuesta de algoritmos uno de ellos es wayCluster desarrollado en

el laboratorio GRIAS de la línea KDD y el otro es K-Frequency que presenta una forma de particular de formar grupos partiendo del algoritmo k-histogram.

3.1 CLUSTERING PARTICIONAL

Un algoritmo de clustering particional obtiene una partición simple de los datos en k grupos [43]. Es decir el usuario determina el número de particiones que desea tener dentro del conjunto de datos. De este tipo de algoritmos el más representativo y básico es el algoritmo k-means que a pesar de ser diseñado exclusivamente para datos numéricos ha tenido algunas modificaciones que le permiten trabajar con datos categóricos, de este algoritmo parten la mayoría de algoritmos que pertenecen a esta técnica. También se implemento el algoritmo k-prototype que es un hibrido entre el algoritmo k-means y k-mode y fue diseñado exclusivamente para conjuntos de datos mixtos.

3.1.1 Algoritmo K-Means

Es uno de los algoritmos más populares dentro de las técnicas de clustering particional y en general de todas las técnicas de clustering. Se define de la siguiente manera: dado un conjunto D = (X_1,. . . , X_n) de n objetos de datos numéricos, un número natural k ≤ n, y una distancia medida d, el algoritmo k-means tiene por objeto la búsqueda de una partición C de D en k no vacía disjunta de grupos C_1,. . . , C_k Con $C_i \cap C_j = \Phi$, y $U^k_{i=1}$ C_i = D tal que el importe global cuadrado de la distancia entre los datos y sus objetos de centros de grupo se reduzcan al mínimo. Matemáticamente, si el uso indicador de variables $w_{i,l}$, que tienen el valor 1 si el objeto X_i es del grupo C_l, y 0 en caso contrario, entonces el problema puede ser declaró en términos de una limitada optimización no lineal problema de la siguiente manera:

Minimizar el valor de distancia de la formula (14).

$$P(W, Q) = \sum_{l=1}^{k} \sum_{i=1}^{n} w_{i,l} d(X_i, Q_l) \qquad (14)$$

Sujeto a

$$\sum_{l=1}^{k} w_{i,l} = 1, \quad 1 \le i \le n, \qquad (15)$$

$$w_{i,l} \in \{0, 1\}, \quad 1 \le i \le n, 1 \le l \le k,$$

Donde W = $[w_{i,l}]_{n \times k}$ es una partición de la matriz, Q = (Q_1,. . . , Q_K) es un conjunto de centros de grupo, y d (.,.). Como es bien sabido, el método habitual hacia la optimización de P en la fórmula 14 sujeto a la restricción de la fórmula 15, es utilizar parcial para la optimización de Q y W. Esto es, en primer lugar fijar Q y encontrar las condiciones necesarias para reducir W al mínimo P. Entonces, fijar y reducir al W mínimo P de acuerdo con Q. Básicamente, el algoritmo k-means itera a través de un proceso de tres pasos hasta que P (W, Q) converge a algunos locales mínimos [64]:

También se muestra en [37][39] que el método k-means puede extenderse a datos categóricos mediante el uso de una simple adecuación a distancia para la medida categórica de objetos con una votación mayoritaria para definir estrategia el "centros de grupo" que se llama modos de transporte. En concreto, la distancia entre dos objetos categórico X_1, X_2 E D, con X1 = ($X_{1,1}$, ..., $X_{1,m}$) y X_2 = ($x_{2,1}$,...,$x_{2,m}$), consistente en categórico de valores sólo puede ser definida como se muestra en la función de similitud.

K-means de Forgy

El método de Forgy se propuso en 1965 y constituye la aproximación más simple al clustering particional. Utiliza el concepto de centroide. El centroide de un clúster se define como el punto equidistante de los objetos pertenecientes a dicho clúster. El algoritmo de Forgy [42], comienza con cualquier configuración inicial, entendiéndose por la misma un conjunto de *k* centroides escogidos bien al azar o bien con una estrategia en la cual dichos centroides se encuentren lo suficientemente separados unos de otros en el espacio *n*-dimensional o por medio de una partición del conjunto de objetos en *k* clústeres, como se ve en la figura 22 en el literal (a) donde están los centroides con forma de cruz.

En el caso de que se haya comenzado por un conjunto de *k* centroides, se obtiene la partición correspondiente sin más que asignar cada objeto que se pretende clasificar al centroide más cercano, calculando la distancia de cada objeto con cada centroide como lo muestra el literal (b) de la figura, mostrando la distancia con una línea roja. Una vez que se haya llevado a cabo la asignación de todos los objetos a clasificar, se debe computar los *k* nuevos centroides, como en el literal (c). Dichos *k* nuevos centroides se utilizarán de nuevo como *atractores*, es decir se asignara a cada uno de los *N* objetos que pretendemos clasificar al centroide más cercano, y volveremos a recalcular los centroides cuando se hayan llevado a cabo la asignación de todos los objetos. La alternación de los 2 pasos anteriores se mantendría hasta que un determinado criterio de parada se verifique. Los criterios de parada pueden ser un número máximo de iteraciones, el que no se produzcan cambios en los clústeres de dos iteraciones sucesivas, el que los nuevos centroides disten de los centroides obtenidos en la iteración previa menos que una

determinada distancia. Generalmente se obtienen resultados diferentes en cada ejecución (ver figura 23).

Figura 22: Procedimiento del algoritmo kmeans

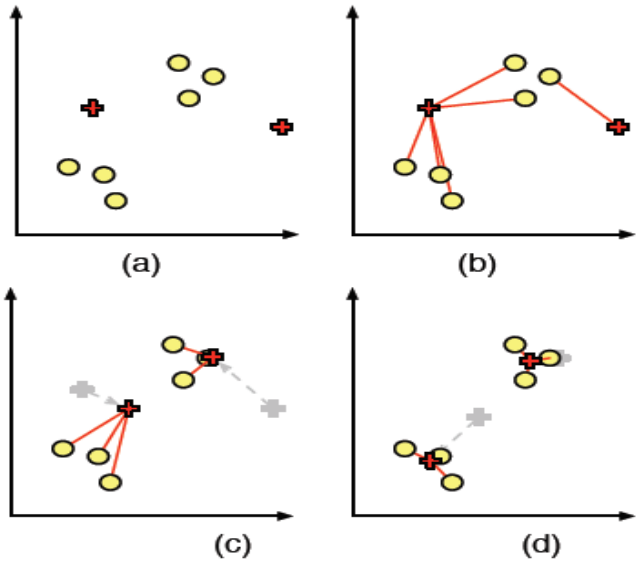


Figura 23: Pasos básicos algoritmo K-means de Forgy

Algoritmo k–means de Forgy

1. Seleccione k objetos semillas de forma aleatoria
2. **Repita**:
   3. calcular la distancia de todos los objetos con los centroides
   4. Asignar cada objeto al cluster con el cual sea más similar a su centroide, es decir que tenga la menor distancia
   5. Calcular de nuevo los centroides de cada cluter, es decir calcular el valor medio de los objetos para cada cluster.
6. **Hasta** cuando no se produzcan cambios (Convergencia) o cuando se determine un número de iteraciones.

Algoritmo k-means de McQueen

Creado por Mac Queen 1967[42]. Es uno de los más simples, conocidos y aplicados algoritmos de agrupamiento, sigue una forma fácil y simple para dividir una base de datos dada en k grupos fijados a priori. El algoritmo propuesto por McQueen comienza considerando los *k* primeros elementos del dataset como los *k* centroides iníciales, o dicho de otra forma equivalente como conglomerados con un único elemento. A continuación, y siguiendo el orden establecido en el dataset,

cada uno de los objetos se va asignando al conglomerado con centroide más próximo, con la característica de que al efectuar cada asignación se recalculan las coordenadas del nuevo centroide. Esta característica es la que básicamente distingue al método de McQueen del método de Forgy. Finalmente, una vez asignados todos los objetos, se calculan los centroides para cada uno de los conglomerados y se reasigna cada objeto al centroide más cercano sin que en este paso se lleve a cabo una recalculación del centroide para cada asignación. Los pasos anteriores se iteran hasta que se verifique un determinado criterio de parada igual que el método Forgy. Es importante tener en cuenta que el algoritmo de McQueen es sensible al orden con el que se encuentran los objetos en el dataset de casos, y fundamentalmente es sensible a los objetos que se encuentran en las *K* primeras posiciones (ver figura 24).

Figura 24: pasos del algoritmo kmeans.

Algoritmo k-means de McQueen

1. Seleccione k primeros objetos como semillas
2. **Repita**:
   2.1 calcular la distancia de todos los objetos con los centroides
   2.2 Asignar cada objeto al cluster con el cual sea más similar a su centroide, es decir que tenga la menor distancia
   2.3 Calcular de nuevo los centroides de cada cluter, es decir calcular el valor medio de los objetos para cada cluster.
3. **Hasta** cuando no se produzcan cambios (Convergencia) o cuando se determine un número de iteraciones.

Ejemplo 1: algoritmo kmeans con el conjunto de datos empleados.

El algoritmo kmeans como se menciono anteriormente tiene dos variantes, la de McQueen y la de Forgy, por tratarse de un ejemplo se desarrollara el método McQueen utilizando el conjunto de datos empleados (ver tabla 1), en el cual se encuentran atributos numéricos y categóricos, todo con el fin de mirar el funcionamiento de los algoritmos.

Tabla 1: Conjunto de datos empleados

Objeto	Casado	Coche	Hijos	Antigüedad	Sexo
o0	Sí	No	0	3	H
o1	Sí	No	3	2	M
o2	Si	Si	2	1	H
o3	No	No	1	1	M
o4	No	Sí	1	3	M
o5	Si	Si	1	2	H

Paso 1: Seleccionar el valor de k = 2, y las semillas serán los k primeros registros. Como se muestra en la tabla 2.

Tabla 2: Semillas del conjunto de datos empleados para el algoritmo kmeans

	Casado	**Coche**	**Hijos**	**Antigüedad**	**Sexo**
Clúster 0	Sí	No	0	3	H
Clúster 1	Sí	No	3	2	M

Paso 2: calcular la distancia entre los objetos y las k semillas, para ellos se utiliza la distancia euclidiana para los atributos de tipo numérico y la distancia hamming para los atributos categóricos, uniendo este tipo de funciones utilizando la formula (13). A continuación se presenta la forma como calcular la distancia entre el objeto o0 y la semilla 1.

$$d(o0, sem1) = ((0 - 3)^2 + (3 - 2)^2)^{½} + (0 + 0 + 1)$$
$$d(o0, sem1) = (9 + 1)^{½} + (1)$$
$$d(o0, sem1) = (3.162) + (1)$$
$$d(o0, sem1) = 4.162$$

De la misma forma como se calculan las demás distancias entre semillas y objetos obteniendo como resultado lo presentado en la tabla 3.

Tabla 3: Distancia entre objetos y semillas tomadas por el algoritmo kmeans

	Distancia semilla 0	**Distancia semilla 1**
o0	0	4.162
o1	4.162	0
o2	3.828	3.414
o3	4.236	3.236
o4	4	4.236
o5	2.4142	4

Paso 3: Asignar cada objeto al clúster con el cual sea más similar a su centroide, teniendo en cuenta la menor distancia presentada en la tabla 3 los clúster formados son clúster 0 = (o0, o4, o5) y el clúster 1 = (o1, o2, o3).

Paso 4: Calcular los centroides de cada clúster, para ello se toma la media en los atributo numéricos y la moda en los atributos categóricos, como se presenta a continuación para el clúster 0, tomando la tabla 1 en donde se encuentra los objetos del conjunto de datos y utilizando la asignación de clústeres mencionada en el anterior paso, se tiene que como el primer atributo es “casado” de tipo

categórico, para este caso se debe calcular la moda, la moda es el valor que más se repite, para el caso del clúster 0 en dicho atributo se tiene que el valor de Si tiene 2 repeticiones y el valor de No 1, es decir la moda para el caso del atributo "casado" seria Si, para el caso del atributo "coche" que es de tipo categórico, se tiene 2 valores por No y 1 por Si, entonces para el atributo "Coche" la moda seria No, para el atributo "Hijos" se debe calcular la media aritmética, ya que es de tipo numérico, para ello se debe sumar todos los valores que se encuentren en dentro del clúster y dividirlo entre el numero de objetos pertenecientes a él así 0+1+1 = 2 divido entre 3 que es el numero de objetos del clúster 0, el resultado es 0.667 que sería la media, por esta razón se denomina a este algoritmo k-means o k-medias. Para el caso del atributo antigüedad se hace lo mismo se toman los calores de los objetos y se suman 3+3+2 = 8 dividido otra vez entre 3, el resultado es 2.667, y por último el atributo "Sexo" de tipo categórico el cual tiene los siguientes valores H = 2 y M = 1, la moda en este caso será el valor H. ya siendo aplicado este concepto a todos los clústeres formados se tiene como resultado los centroides presentados en la tabla 4.

Tabla 4: Centroides parciales paso 4 de kmeans en empleados.

	Casado	**Coche**	**Hijos**	**Antigüedad**	**Sexo**
Clúster 0	Si	Si	0.667	2.667	H
Clúster 1	Si	No	2	1.333	M

Paso 5: Calcular la distancia entre los centroides de los k clústeres y los objetos. El resultado de las distancias se presenta en la tabla 5.

Tabla 5: Distancia entre objetos y centroides de la tabla 4 en el algoritmo kmeans

Objeto	**Distancia clúster 0**	**Distancia clúster 1**
o0	1.745	3.603
o1	4.427	1.202
o2	2.134	2.333
o3	4.700	2.054
o4	2.471	3.944
o5	0.745	3.202

Paso 6: Asignar cada objeto al clúster con el cual sea más similar a su centroide, es decir que tenga la menor distancia. Teniendo en cuenta la tabla 5 los clúster formados son clúster 0 = (o0, o2, 04, o5) y el clúster 1 = (o1, o3).

Paso 7: Calcular los centroides de cada clúster, es decir calcular el valor medio de los objetos para cada clúster y las modas respectivamente. Teniendo como resultado los centroides presentados en la tabla 6.

Tabla 6: Centroides parciales paso 7 de kmeans en empleados.

	Casado	Coche	Hijos	Antigüedad	Sexo
Clúster 0	Sí	Si	1	2.25	H
Clúster 1	Si	No	2	1.5	M

Paso 8: Calcular la distancia entre los centroides de los k clústeres y los objetos.

Tabla 7: Distancia entre objetos y centroides de la tabla 6 en el algoritmo kmeans

Objeto	Distancia clúster 0	Distancia clúster 1
o0	2.25	3.5
o1	4.016	1.118
o2	1.601	2.5
o3	4.25	2.118
o4	2.75	3.803
o5	0.25	3.118

Paso 9: Asignar cada objeto al clúster con el cual sea más similar a su centroide, es decir que tenga la menor distancia. Teniendo en cuenta la tabla 7 los clúster formados son clúster 0 = (o0, o1, 03) y el clúster 2 = (o2, o4). Los clústeres creados no tuvieron cambio con respecto al paso 6, entonces se llego al criterio de convergencia y teniendo como centroides los establecidos en la tabla 6. El resultado final se presenta en la tabla 8.

Tabla 8: Resultado final del algoritmo kmeans en empleados con k=2.

Objeto	Casado	Coche	Hijos	Antigüedad	Sexo	Clúster
o0	Sí	No	0	3	H	0
o1	Sí	No	3	2	M	1
o2	Si	Si	2	1	H	0
o3	No	No	1	1	M	1
o4	No	Sí	1	3	M	0
o5	Si	Si	1	2	H	0

3.1.2 Algoritmo k-prototype

Tradicionalmente, métodos numéricos de agrupación se han visto en oposición a la agrupación conceptual desarrollando métodos en la Inteligencia Artificial. Hacer hincapié en técnicas numéricas, la determinación de grupos homogéneos de acuerdo a algunas medidas de similitud, de bajo nivel, buscando descripciones de las agrupaciones, mientras que un enfoque conceptual es más un tratamiento de alto nivel (es decir, más comprensible) de descripciones de las clases [40]. La mayoría de los trabajos en clustering se han centrado sobre datos numéricos cuya inherentes propiedades geométricas puede ser explotado a distancia, naturalmente, definiendo funciones entre los puntos de datos. Sin embargo, las aplicaciones de minería de datos a menudo a muchos conjuntos de datos que también contienen atributos categóricos en que las funciones se encuentren una distancia Naturalmente, no se define.

El clustering de datos con atributos categóricos ha llamado a algunos la atención. Como es bien sabido, la agrupación k-means ha sido una técnica muy popular para el particionado de grandes conjuntos de datos con atributos numéricos. Ralambondrainy propone un híbrido simbólico-numérico método que integra una versión ampliada de la k-means algoritmo de determinación de grupo y un algoritmo de complemento de caracterización conceptual de descripción de clúster.

Sin embargo, la conversión de los atributos categóricos en atributos binarios (Ralambondrainy) en el enfoque hace que la técnica propuesta de hacer frente a la cada vez mayor de ambos computacional y en el espacio si los costos de los atributos categóricos tienen muchas categorías. Además, los valores reales entre 0 y 1 en representación del grupo de medios no indican las características de los clúster. Recientemente, [37][39] se propone el algoritmo k-modes que aborda el problema de la agrupación de grandes conjuntos de datos categóricos en la minería de datos.

El k-modes se extiende el algoritmo de k-means algoritmo mediante el uso de una simple medida de disimilitud a los correspondientes atributos categóricos de objetos, la moda en lugar de los medios a las agrupaciones, y un método basado en frecuencias para actualizar los modos de agrupamiento en el proceso de reducir al mínimo la función de agrupación de costos. Además, Huang también combinado el k-modes con el algoritmo de k-means algoritmo resultante en el llamado K-prototypes algoritmo para agrupar los objetos descritos por atributos numéricos, categóricos o mixtos.

El algoritmo k-prototype combina el algoritmo k-means y k-mode [38]. Esto se logra mediante la definición de una nueva función de la distancia que combina tanto la distancia euclidiana al cuadrado y la medida de disimilitud de concordancia simple como se representa en la formula (13).

$$d(x,y) = \sum_{j=0}^{pn} (x_j - y_j)^2 + \sum_{j=pn+1}^{m} \delta(x_j, y_j) \quad (13)$$

Donde en la primera formula es la suma de las distancias euclidianas de los pn atributos numéricos y la segunda fórmula es la suma de la disimilitud de los $m - pn$ atributos categóricos, siendo m el número total de atributos.

Como se ha visto, en la aplicación del método k-means para objetos categóricos, se encuentran dos problemas principales, a saber, la formación de centros de grupo y el cálculo de disimilitud entre los objetos y los centros de grupo. Estos problemas han sido completamente resueltos en el k-modes algoritmo mediante el uso de la simple equiparación de medida de disimilitud para los datos categóricos en lugar de la medida euclidiana de distancia, y la sustitución de los medios de las agrupaciones por los modos de transporte.

Formación de centros de clúster

Como las operaciones aritméticas están completamente ausentes en el establecimiento categórico de los objetos, se utiliza el producto cartesiano y las operaciones de la Unión para la formación de “centros de grupo” sobre la base de la noción de los medios y en el establecimiento. En particular, y además se sustituye en la multiplicación por la Unión y el producto cartesiana, respectivamente, para datos categórica en la definición de la noción de representantes de agrupaciones.

Habida cuenta de un grupo C = (X_1,..., X_p) de objetos categóricos, con

$$Xi = (x_{i,1},..., X_{i,m}), 1 \leq i \leq p$$

Denotan por el conjunto que se formó a partir de los valores categóricos $x_{1,j}$,. . . , $X_{p,j}$. Por ejemplo, el conjunto se formó a partir de los valores a, b, a, c es {a, b, c}.

A continuación, el representante de C se define por Q = (q_1,..., q_m), con

$$q_j = \{(C_j, f_{cj}) \mid C_j \text{ E } D_j\},$$

Donde f_{cj} es la frecuencia relativa de la categoría dentro de c_j, es decir, C,i,e.., f_{cj} = n_{cj} / p donde n_{cj} es el número de objetos en C que tengan categoría en $_{CJ}$ atributo A_j. Formalmente, cada qj puede ser visto como un conjunto difuso de Dj con la pertenencia grados de elementos que se definen por sus frecuencias relativas en el seno del grupo. Valdría la pena señalar que si aplicamos la definición anterior

de los representantes. Para objetos de datos numéricos con la sustitución de la unión y el producto cartesiano por sumas y multiplicaciones, respectivamente, se obtendrá el concepto de centros de grupo de medios.

Pasos algoritmo k-prototype

Con las modificaciones que acaba de mencionar anteriormente, ahora se formula el problema de agrupar los datos categóricos como un problema de particionamiento en forma similar al algoritmo k-mode. Suponiendo que se tiene un conjunto de datos D = $\{X_1,..., X_{n}\}$ de objetos perteneciente a un grupo, en la que cada objeto $X_i = (x_{i,1},..., X_{i,m})$, $1 \leq i \leq n$ es descrito por m atributos categóricos.

Entonces, el problema puede ser matemáticamente como se ve en la formula (18)

Minimizar $$P(W, Q) = \sum_{l=1}^{k} \sum_{i=1}^{n} w_{i,l} d(X_i, Q_l), \qquad (18)$$

Sujeto a:

$$\sum_{l=1}^{k} w_{i,l} = 1, \quad 1 \leq i \leq n, \qquad (19)$$

$$w_{i,l} \in \{0, 1\}, \quad 1 \leq i \leq n, 1 \leq l \leq k,$$

Donde $W = [w_{i,l}]_{n \times k}$ es una partición de la matriz, $Q = (Q_1,..., Q_K)$ es el conjunto de los representantes, y $d(X_i, Q_L)$ es la diferencia entre objeto y X_i representante Q_l definido por la formula de distancia.

Figura 25: Pasos algoritmo K-Prototype.

Pasos Básicos Algoritmo K-Prototype

1. Iniciar k partiones de forma aleatoria
2. Calcular los centroides para cada clúster.
3. **Repita**:
   - 3.1 Calcular la distancia de todos los objetos con los representantes
   - 3.2 Asignar cada objeto al cluster con el cual sea más similar a su representante, es decir que tenga la menor distancia
   - 3.3 Al asignar un objeto al nuevo clúster, y recalcular los representantes de los dos grupos
   - 3.4 Calcular de nuevo los centroides de cada clúster, es decir calcular el valor medio de los objetos para cada cluster.
4. **Hasta** cuando no se produzcan cambios (Convergencia) o cuando se determine un número de iteraciones.

También se debe señalar que la definición de los representantes de las agrupaciones en la propuesta técnica, se basa en la noción de los medios, es decir, la optimización de la solución del problema numérico correspondiente, reduciendo así a una forma parcial el problema, teniendo que P1 [39], de W, tal como se especifica en el mencionado algoritmo. Según la fórmula de distancia larga se dice que:

$$d(X_i, Q_l) = \sum_{j=1}^{m} (1 - f_{x_{i,j}}), \qquad (20)$$

Donde $f_{xi,j}$ es la frecuencia relativa de la categoría $x_{i,j.}$ en el grupo C_l. Así, en el algoritmo propuesto objeto X_i se asignarán al grupo C_l a fin de que las categorías de X_i sean más probables que constituyen a la moda de C_l relacionados a los demás grupos. Hay que tener en cuenta que, por definición, todas las posibles modas de cada atributo dentro del clúster, son las que tienen se tienen en cuenta en el algoritmo.

Ejemplo 2: algoritmo k-prototype con el conjunto de datos empleados.

El algoritmo k-prototype por tratarse de un ejemplo se desarrollara calculando la distancia euclidiana para los atributos numéricos y utilizando la disimilitud de hammig para los categóricos con k=2, en el conjunto de datos empleados (ver tabla 1).

Paso 1: hacer la asignación de clúster a los objetos de forma aleatoria. Como se muestra en la tabla 9.

Tabla 9: Resultado de la asignación aleatoria de clúster a los objetos.

Objeto	Casado	Coche	Hijos	Antigüedad	Sexo	Clúster
o0	Sí	No	0	3	H	0
o1	Sí	No	3	2	M	1
o2	Si	Si	2	1	H	1
o3	No	No	1	1	M	1
o4	No	Sí	1	3	M	0
o5	Si	Si	1	2	H	1

Paso 2: calcular los centroides para los clústeres formados.

Tabla 10: Centroide de la agrupación aleatoria de k-prototype en empleados

	Casado	Coche	Hijos	Antigüedad	Sexo
Clúster 0	Sí	No	0.5	3	H
Clúster 1	Si	No	1.75	1.5	M

Paso 3: calcular la distancia entre los centroides y los clúster y asignar el objeto al clúster con menor distancia y si cambia algún objeto de clúster actualizar los centroides.

- Objeto 0:
  - d(o0, cluster 0) = 0.5
  - d(o0, clúster 1) = 3.305

Se conserva o0 en clúster 0.

- Objeto 1:
  - d(o1, cluster 0) = 3.693
  - d(o1, clúster 1) = 1.346

Se conserva o1 en clúster 1

- Objeto 2:
  - d(o2, cluster 0) = 3.5
  - d(o2, clúster 1) = 2.559

Se conserva o2 en clúster 1

- Objeto 3:
  - d(o3, cluster 0) = 4.062
  - d(o3, clúster 1) = 1.901

Se conserva o3 en clúster 1

- Objeto 4:
  - d(o4, cluster 0) = 3.5
  - d(o4, clúster 1) = 3.667

Se conserva o4 en clúster 0

- Objeto 5:
  - d(o5, clúster 5) = 2.118
  - d(o5, clúster 5) = 2.901

Se cambia o5 a clúster 0

Se modificaron los miembros de los clúster 0 = {o0, o4. o5}; y clúster 1 = {o1, o2, o3}, se debe volver a calcular los centroides y el resultado se presenta en la tabla 11.

Tabla 11: Nuevos centroides de kprototype en empleados se cambio del objeto o5 del clúster 1 al clúster 0.

	Casado	Coche	Hijos	Antigüedad	Sexo
Clúster 0	Sí	Si	0.667	2.667	H
Clúster 1	Si	No	2	1.333	M

Paso 4: calcular la distancia entre los centroides y los clúster y asignar el objeto al clúster con menor distancia y si cambia algún objeto de clúster actualizar los centroides.

- Objeto 0:
  - d(o0, cluster 0) = 1.745
  - d(o0, clúster 1) = 3.603

Se conserva o0 en clúster 0.

- Objeto 1:
  - d(o1, cluster 0) = 4.427
  - d(o1, clúster 1) = 1.202

Se conserva o1 en clúster 1.

- Objeto 2:
  - d(o2, cluster 0) = 2.134
  - d(o2, clúster 1) = 2.333

Se cambia o2 a clúster 0.

Se modificaron los miembros de los clúster 0 = {o0, o2, o4, o5}; y clúster 1 = {o1, o3}, se debe volver a calcular los centroides y el resultado se presenta en la tabla 12.

Tabla 12: Nuevos centroides de k-prototype en empleados se cambio del objeto o2 del clúster 1 al clúster 0.

	Casado	Coche	Hijos	Antigüedad	Sexo
Clúster 0	Sí	Si	1	2.25	H
Clúster 1	Si	No	2	1.5	M

Paso 5: se sigue calculando la distancia entre los objetos restantes con los nuevos centroides.

- Objeto 3:
  - d(o3, clúster 0) = 4.25
  - d(o3, clúster 1) = 2.118

Se conserva o3 en clúster 0.

- Objeto 4:
  - d(o4, clúster 0) = 2.75
  - d(o4, clúster 1) = 3.803

Se conserva o4 en clúster 0.

- Objeto 5:
  - d(o5, clúster 0) = 0.25
  - d(o5, clúster 1) = 3.118

Se conserva o5 en clúster 0.

Paso 6: como hubo cambios en la formación de los clústeres se debe volver a encontrar la distancia entre los centroides de la tabla 12 y los objetos.

- Objeto 0:
  - d(o0, cluster 0) = 2.25
  - d(o0, clúster 1) = 3.5

Se conserva o0 en clúster 0.

- Objeto 1:
  - d(o1, cluster 0) = 4.016
  - d(o1, clúster 1) = 1.118

Se conserva o1 en clúster 1.

- Objeto 2:
  - d(o2, cluster 0) = 1.601
  - d(o2, clúster 1) = 2.5

Se conserva o2 en clúster 0.

- Objeto 3:
  - d(o3, cluster 0) = 4.25
  - d(o3, clúster 1) = 2.118

Se conserva o3 en clúster 1

- Objeto 4:
  - d(o4, cluster 0) = 2.75
  - d(o4, clúster 1) = 3.803

Se conserva o4 en clúster 0.

- Objeto 5:
  - d(o5, clúster 5) = 0.25
  - d(o5, clúster 5) = 3.118

Se cambia o5 a clúster 0

Como no hubo cambios en los clústeres, se ha llegado a un criterio de convergencia. El resultado final se presenta en la tabla 13.

Tabla 13: Resultado final del algoritmo k-prototype en empleados con k=2.

Objeto	Casado	Coche	Hijos	Antigüedad	Sexo	Clúster
o0	Sí	No	0	3	H	0
o1	Sí	No	3	2	M	1
o2	Si	Si	2	1	H	0
o3	No	No	1	1	M	1
o4	No	Sí	1	3	M	0
o5	Si	Si	1	2	H	0

3.1.3 Algoritmo wayCluster

Uno de los mayores inconvenientes en la aplicación de técnicas de clustering de tipo particional es definir el número de grupos que se desea formar en un conjunto de datos buscando una solución a este inconveniente nació una propuesta de agrupación teniendo en cuenta que uno de los objetivos del clustering es maximizar la similitud entre objetos de un mismo clúster, es posible conseguir este objetivo agrupando en un mismo clúster los objetos que tienen entre sí la menor distancia, es decir un objeto se unirá a otro objeto con el cual tenga la menor distancia dentro del conjunto de datos. Los datos que no formen parte de algún clúster se interpreta como dato distante o como nuevo clúster, creando así clúster con distancias bajas entre sus elementos y maximizando la distancia entre los clústeres creados. Los pasos del algoritmo los están plasmados en la figura 26.

Figura 26: Pasos algoritmo wayCluster.

Pasos Algoritmo wayCluster

1. Calcular la matriz de distancia
2. **Mientras**: haya objetos i
   - 2.1: Encontrar la menor distancia el objeto i, entre sus distancias.
   - 2.2: **SI** el objeto no ha sido asignado a algún clúster
     - 2.2.1 ENTONCES crear un nuevo clúster y asignarlo el objeto i.
   - 2.3: **Mientras:** haya objetos j
     - 2.3.1 **SI** la distancia del objeto j es igual a distancia menor y el objeto j y la distancia menor es menor que la distancia del cluster a la que fue asignado el objeto j.
     - 2.2.1.1 **Entonces:** Asignar el objeto j al cluster que pertenece el objeto i y actualizar su valor de distancia asignada.
   - **Fin Mientras.**

   Fin Mientras.
3. Calcular los centroides de cada clúster.
4. Finalizar.

Ejemplo 3: algoritmo wayCluster con el conjunto de datos empleados.

Se aplicara el algoritmo wayCluster en el conjunto de datos empleados (ver tabla 1) no tiene ningún parámetro para establecer el número de clústeres a formar lo único que necesita es establecer la medidas de distancia y disimilitud. Para este ejemplo se utilizara la distancia euclidiana y la distancia hammig respectivamente.

Paso 1: calcular la matriz de distancias entre los objetos del conjunto de datos.

Tabla 14: Matriz de distancia entre objetos del conjunto de datos empleados.

	o0	**o1**	**o2**	**o3**	**o4**	**o5**
o0	0	4.162	3.828	4.236	4	2.414
o1	4.162	0	3.414	3.236	4.236	4
o2	3.828	3.414	0	4	4.36	1.414
o3	4.236	3.236	4	0	3	4
o4	4	4.236	4.236	3	0	3
o5	2.414	4	1.414	4	3	0

Paso 2: se encuentra la distancia menor del objeto o0 exceptuando la distancia con el mismo, que seria 2.414 con o5.

Tabla 15: Iteración numero 1 wayCluster, menor distancia o0

	o0	**o1**	**o2**	**o3**	**o4**	**o5**
o0	0	4.162	3.828	4.236	4	2.414
o1	4.162	0	3.414	3.236	4.236	4
o2	3.828	3.414	0	4	4.36	1.414
o3	4.236	3.236	4	0	3	4
o4	4	4.236	4.236	3	0	3
o5	2.414	4	1.414	4	3	0

Paso 3: se mira si ya fue asignado o0 a algún clúster, como no es así, se crea el clúster 0 y se asigna a el objeto o0. (ver tabla 15).

Paso 4: comparo la distancia menor de o0 con sus demás distancias y miro cual es igual. Observando la tabla 15 el valor menor es igual al valor del objeto o5, entonces o5 entra a formar parte del clúster 0, se determina que no hay mas objetos similares a o0 teniendo como resultado la tabla 16 y se continua.

Tabla 16: Resultado primera iteración wayCluster en empleados.

	o0	**o1**	**o2**	**o3**	**o4**	**o5**
o0	0	4.162	3.828	4.236	4	2.414
o1	4.162	0	3.414	3.236	4.236	4
o2	3.828	3.414	0	4	4.36	1.414
o3	4.236	3.236	4	0	3	4
o4	4	4.236	4.236	3	0	3
o5	2.414	4	1.414	4	3	0

Paso 5: pasamos a evaluar el objeto o1, se encuentra la distancia menor de o1 en tabla 17, concluyendo que a menor distancia es 3.236 con o3.

Tabla 17: Iteración numero 2 wayCluster, menor distancia o1

	o0	**o1**	**o2**	**o3**	**o4**	**o5**
o0	0	4.162	3.828	4.236	4	2.414
o1	4.162	0	3.414	3.236	4.236	4
o2	3.828	3.414	0	4	4.36	1.414
o3	4.236	3.236	4	0	3	4
o4	4	4.236	4.236	3	0	3
o5	2.414	4	1.414	4	3	0

Paso 6: se mira si ya fue asignado o1 a algún clúster, como no es así, se crea el clúster 1 y se asigna a el objeto o1.

Paso 7: se compara la distancia menor de o1 con sus demás distancias y miro cual es igual. Observando la tabla 17 el valor menor es igual al valor del objeto o3, entonces o3 entra a formar parte del clúster 1, se determina que no hay mas objetos similares a o1 (ver tabla 18) y se continua.

Tabla 18: Resultado después de Iteración numero 2 wayCluster.

	o0	**o1**	**o2**	**o3**	**o4**	**o5**
o0	0	4.162	3.828	4.236	4	2.414
o1	4.162	0	3.414	3.236	4.236	4
o2	3.828	3.414	0	4	4.36	1.414
o3	4.236	3.236	4	0	3	4
o4	4	4.236	4.236	3	0	3
o5	2.414	4	1.414	4	3	0

Paso 8: pasamos a evaluar el objeto o2, se encuentra la distancia menor de o2 en tabla 19, concluyendo que a menor distancia es 1.414 con o5.

Tabla 19: Iteración numero 3 wayCluster, menor distancia o2.

	o0	**o1**	**o2**	**o3**	**o4**	**o5**
o0	0	4.162	3.828	4.236	4	2.414
o1	4.162	0	3.414	3.236	4.236	4
o2	3.828	3.414	0	4	4.36	1.414
o3	4.236	3.236	4	0	3	4
o4	4	4.236	4.236	3	0	3
o5	2.414	4	1.414	4	3	0

Paso 9: se mira si ya fue asignado o2 a algún clúster, como no es así, se crea el clúster 2 y se asigna a el objeto o2.

Paso 10: se compara la distancia menor de o2 con sus demás distancias y miro cual es igual. Observando la tabla 19 el valor menor es igual al valor del objeto o5, entonces o5 entra a formar parte del clúster 2, se determina que no hay mas objetos similares a o2 (Ver tabla 20) y se continua.

Tabla 20: Resultado iteración numero 2 wayCluster.

	o0	**o1**	**o2**	**o3**	**o4**	**o5**
o0	0	4.162	3.828	4.236	4	2.414
o1	4.162	0	3.414	3.236	4.236	4
o2	3.828	3.414	0	4	4.36	1.414
o3	4.236	3.236	4	0	3	4
o4	4	4.236	4.236	3	0	3
o5	2.414	4	1.414	4	3	0

Paso 11: se pasa a evaluar el objeto o3, se encuentra la distancia menor de o3 en tabla 21, concluyendo que la menor distancia es 3 con o4.

Tabla 21: Iteración numero 4 wayCluster, menor distancia o3

	o0	**o1**	**o2**	**o3**	**o4**	**o5**
o0	0	4.162	3.828	4.236	4	2.414
o1	4.162	0	3.414	3.236	4.236	4
o2	3.828	3.414	0	4	4.36	1.414
o3	4.236	3.236	4	0	3	4
o4	4	4.236	4.236	3	0	3
o5	2.414	4	1.414	4	3	0

Paso 12: se mira si ya fue asignado o3 a algún clúster, como si es así, se va a comparar con los otros objetos.

Paso 13: se compara la distancia menor de o3 con sus demás distancias y miro cual es igual. Observando la tabla 21 el valor menor es igual al valor del objeto o4, entonces o4 entra a formar parte del clúster 1 ya que es el clúster al que forma parte o3, se determina que no hay mas objetos similares a o3 (ver tabla 22) y se continua.

Tabla 22: Resultado iteración numero 4 wayCluster.

	o0	**o1**	**o2**	**o3**	**o4**	**o5**
o0	0	4.162	3.828	4.236	4	2.414
o1	4.162	0	3.414	3.236	4.236	4
o2	3.828	3.414	0	4	4.36	1.414
o3	4.236	3.236	4	0	3	4
o4	4	4.236	4.236	3	0	3
o5	2.414	4	1.414	4	3	0

Paso 11: se procede a evaluar el objeto o4, se encuentra la distancia menor de o4 en tabla 23, concluyendo que a menor distancia es 3 con o3 y o5.

Tabla 23: Iteración numero 5 wayCluster, menor distancia o4.

	o0	**o1**	**o2**	**o3**	**o4**	**o5**
o0	0	4.162	3.828	4.236	4	2.414
o1	4.162	0	3.414	3.236	4.236	4
o2	3.828	3.414	0	4	4.36	1.414
o3	4.236	3.236	4	0	3	4
o4	4	4.236	4.236	3	0	3
o5	2.414	4	1.414	4	3	0

Paso 12: se mira si ya fue asignado o4 a algún clúster, como si es así, se va a comparar con los otros objetos.

Paso 13: se compara la distancia menor de o4 con sus demás distancias y miro cual es igual. Observando la tabla 14 el valor menor es igual al valor del objeto o3 y o5, pero como o4 ya está asignado al clúster 1 entonces o4 se queda en el clúster 1 ya que es el clúster al que forma parte o3, se determina que no hay mas objetos similares a o4 (ver tabla 24) y se continua.

Tabla 24: Resultado iteración numero 5 wayCluster, menor distancia o4.

	o0	**o1**	**o2**	**o3**	**o4**	**o5**
o0	0	4.162	3.828	4.236	4	2.414
o1	4.162	0	3.414	3.236	4.236	4
o2	3.828	3.414	0	4	4.36	1.414
o3	4.236	3.236	4	0	3	4
o4	4	4.236	4.236	3	0	3
o5	2.414	4	1.414	4	3	0

Paso 14: se continua para evaluar el objeto o5, se encuentra la distancia menor de o5 en tabla 25, concluyendo que a menor distancia es 1.414 con o2.

Paso 15: se mira si ya fue asignado o5 a algún clúster, como si es así, se va a comparar con los otros objetos.

Tabla 25: Iteración numero 6 wayCluster, menor distancia o5.

	o0	**o1**	**o2**	**o3**	**o4**	**o5**
o0	0	4.162	3.828	4.236	4	2.414
o1	4.162	0	3.414	3.236	4.236	4
o2	3.828	3.414	0	4	4.36	1.414
o3	4.236	3.236	4	0	3	4
o4	4	4.236	4.236	3	0	3
o5	2.414	4	1.414	4	3	0

Paso 16: se compara la distancia menor de o5 con sus demás distancias y miro cual es igual. Observando la tabla 14 el valor menor es igual al valor del objeto o2, pero como o5 ya está asignado al clúster 2 entonces o5 se queda en el clúster 2 ya que es el clúster al que forma parte o2, se determina que no hay mas objetos similares a o5 (ver tabla 26) y se continua.

Tabla 26: Resultado iteración numero 6 wayCluster, menor distancia o5.

	o0	**o1**	**o2**	**o3**	**o4**	**o5**
o0	0	4.162	3.828	4.236	4	2.414
o1	4.162	0	3.414	3.236	4.236	4
o2	3.828	3.414	0	4	4.36	1.414
o3	4.236	3.236	4	0	3	4
o4	4	4.236	4.236	3	0	3
o5	2.414	4	1.414	4	3	0

Paso 17: como ya no hay objetos que evaluar, entonces se determina que se ha llegado al final. El resultado del algoritmo wayCluster ha formado tres grupos ver tabla 27.

Tabla 27: Resultado final del algoritmo wayCluster en empleados.

Objeto	Casado	Coche	Hijos	Antigüedad	Sexo	Clúster
o0	Sí	No	0	3	H	0
o1	Sí	No	3	2	M	1
o2	Si	Si	2	1	H	2
o3	No	No	1	1	M	1
o4	No	Sí	1	3	M	1
o5	Si	Si	1	2	H	2

3.1.4 Algoritmo K-Frequency.

La mayoría de los algoritmos de clustering se han enfocado en buscar soluciones para tipos de datos solo numéricos o solo categóricos, lo que hace que cuando se tenga un conjunto de datos que dentro de sus atributos se encuentren diferentes tipos de de datos tanto numéricos como categóricos, se deba hacer un análisis de estas variables por separado, perdiendo muchas veces información valiosa de análisis al no tener un análisis complejo, y muchas veces creando respuestas ambiguas. Un algoritmo que se ha destacado por su versatilidad y manejo de conjunto de datos con gran numero de registros ha sido el algoritmos k-means [57].

Con la preocupación en los últimos tiempos de encontrar una respuesta para mejorar la manera de encontrar buenos clústeres en tipos de datos categóricos como k-mode o k-prototype mencionado anteriormente que han buscado nuevas formas de agrupación de tipos de datos categóricos y datos mixtos respectivamente. Un algoritmo denominado k-histogram [20] que es una ampliación del algoritmo k-means a dominios de tipo de datos categóricos mediante la sustitución de la media de los clústeres por histogramas, y las actualizaciones de los histogramas se hacen dinámicamente en el proceso de agrupación, todo con el fin de agrupar los objetos utilizando el histograma, que guarda las frecuencias relativas dentro de los clústeres de los valores de las variables categóricas para describir la estructuración de los atributos categóricos. Estos valores encontrados en el histograma son reemplazados por las funciones de distancia, para entender mejor, a continuación se presentan las notaciones que se tienen en cuenta para la ejecución del algoritmo.

Sea un conjunto de datos D de objetos { X_1, X_2,...,X_n}, descritos por atributos categóricos A_1,….A_m, donde A_i tiene valores categóricos v_1...v_n donde v_i esta denotada por con frecuencia relativa f_i, defiendo el histograma de A_i como el conjunto de pares {(v_i, f_i),…. (v_n, f_n)}, es decir los histogramas del conjunto de datos D se define como H = h_1,…,h_n.

Tomado dos objetos X,Y descritos por m atributos categóricos, la disimilitud medida entre X y Y será definida por el número total de desajustes de los valores correspondientes de los atributos de los objetos. Cuanto menor sea el número de discordancias, más similares serán los dos objetos. Formalmente:

$$d_1(X,Y) = \sum_{j=1}^{m} \delta(x_j, y_j) \qquad (21)$$

Donde:

$$\delta(x_j, y_j) = \begin{cases} 0 & (x_j = y_j) \\ 1 & (x_j \neq y_j) \end{cases} \qquad (22)$$

Siendo el conjunto de datos D = {X_1, X_2,…., X_n} y un objeto Y, la disimilitud medida entre X y Y puede ser definida por el promedio de la suma de los distancias entre Xi y Y.

$$d_2(D,Y) = \frac{\sum_{j=1}^{n} d_1(X_j, Y)}{n} \qquad (23)$$

Si se toma el histograma H = (h_1, h_1, …, H_m) como la representación compacta de el conjunto de datos D, la formula (23) puede ser redefinida por (24).

$$d_3(H,Y) = \frac{\sum_{j=1}^{m} \phi(h_j, y_j)}{n} \qquad (24)$$

Donde:

$$\phi(h_j, y_j) = \sum_{l=1}^{p_j} f_l * \delta(v_l, y_j) \qquad (25)$$

Desde un punto de vista de la eficiencia de la aplicación, la formula (24) puede presentarse en forma de (26).

$$d_4(H,Y) = \frac{\sum_{j=1}^{m} \psi(h_j, y_j)}{n} \qquad (26)$$

Donde:

$$\psi(h_j, y_j) = \sum_{l=1}^{p_j} f_l * (1 - \delta(v_l, y_j)) \qquad (27)$$

La formula (26) puede ser calculada de manera más eficiente para los que solo requieran las frecuencias combinando los atributos pares.

Modificaciones de k-histogram para datos mixtos.

Para lograr que el algoritmo k-histogram logre trabajar con conjuntos de datos mixtos, se propone el algoritmo k-frequency, en el cual se hacen algunos ajustes para poder extender este algoritmo a conjuntos de datos mixtos, cancelando la división de *n* de la fórmula 26, donde n es el numero de atributos, ya que *n* solo promedia las diferencias encontradas entre los atributos. Además para hacer combinación de la diferencia resultante de los atributos categóricos con los histogramas y la distancia encontrada con los atributos de datos numéricos, se propone utilizar la formula (28).

$$d(X_i, Q_l) = \sum_{j=1}^{m} (1 - f_{x_{i,j}}), \qquad (28)$$

Donde X_i es el elemento de conjunto de datos D y Q_l es el representante del clúster *l* y f es la frecuencia en $Q_{l,j}$ del valor de X_i en el atributo j. por ejemplo si en Q_l el atributo j es de tipo categórico con valores a y b y las frecuencias relativas son 0.7 y 0.3 respectivamente y el valor de Xi en el atributo j es a. La disimilitud entre Q_l y X_i en el atributo j aplicando la fórmula 28 es 0.3 correspondiente a la diferencia entre 1 y la frecuencia relativa de a en $Q_{l,j}$. la razón de utilizar la diferencia con 1 es que la frecuencia es inversamente proporcional a la disimilitud, es decir entre más frecuente sea un valor es menos la disimilitud de determinado valor. En la figura 27 se presentan los pasos generales del algoritmo k-frequency.

Además el algoritmo k-frequency en la primera instancia no asigna a los objetos clústeres de forma aleatoria, sino utiliza una función que toma las k primeros

registros como semillas de los clústeres y asigna los demás objetos a los clústeres más cercanos.

Figura 27: Pasos algoritmo k-frequency

Pasos Algoritmo k-frequency

1. Selecciones en valor de k número de clústeres

2. Seleccionar los k primeros objetos como semillas

3. Calcular la distancia entre las semillas y los objetos

4. Asignar el objeto a la semilla más cercana.

5. Calcular los nuevos centroides creando a los atributos categóricos el histograma y en los numéricos la media.

6. **Repita**:
   6.1 Calcular la distancia de todos los objetos con los representantes, utilizando el histograma
   6.2 Asignar cada objeto al clúster con el cual sea más similar a su representante, es decir que tenga la menor distancia
   6.3 Al asignar un objeto al nuevo clúster, y recalcular los representantes de los dos grupos y actualizar los respectivos histogramas.
   6.4 Calcular de nuevo los representantes de cada clúster, es decir calcular el valor medio de los objetos para cada clúster.
   Hasta Cuando no se produzcan cambios (Convergencia) o cuando se determine un número de iteraciones 3. Calcular los centroides de cada clúster.

7. Finalizar.

Ejemplo 4: algoritmo k-frequency con el conjunto de datos empleados.

A continuación se desarrollara un ejemplo del algoritmo k-frequency utilizando la distancia euclidiana para los atributos numéricos y utilizando la disimilitud que se propone para los atributos categóricos y es la esencia de este algoritmo utilizando el valor de k=2, aplicado al conjunto de datos empleados (ver tabla 1).

Paso 1: seleccionar los dos primeros objetos del conjunto de datos como semillas (ver tabla 28).

Tabla 28: Semillas del conjunto de datos empleados para el algoritmo kmeans

	Casado	Coche	Hijos	Antigüedad	Sexo
Clúster 0	Sí	No	0	3	H
Clúster 1	Sí	No	3	2	M

Paso 2: calcular la distancia entre las semillas y los objetos, para esta primera parte se utilizara como medida de disimilitud la proporción de coincidencias para los atributos categóricos (ver capitulo 2.2.2) y para los atributos numéricos la distancia euclidiana (ver tabla 29).

Tabla 29: Distancia entre los objetos y las semillas de los clústeres.

Objeto	Distancia clúster 0	Distancia clúster 1
o0	0	3.496
o1	3.496	0
o2	3.162	2.081
o3	2.903	1.914
o4	2	2.610
o5	1.451	1.869

Paso 3: asignar los objetos al clúster con la menor distancia con la semilla teniendo en cuenta la tabla 29. Los grupos formados serán clúster 0 = {o0, o4, o5} y el clúster 1 = {o1, o2, o3};

Paso 4: calcular los nuevos centroides de los clústeres formados. El resultado se muestra en la tabla 30.

Tabla 30: Nuevos centroides del algoritmo k-frequency en empleados.

	Casado	Coche	Hijos	Antigüedad	Sexo
Clúster 0	Sí	Si	0.667	2.667	H
Clúster 1	Sí	No	2	1.333	M

Paso 5: calcular la distancia entre los objetos y los nuevos centroides (ver tabla 30) utilizando la formula 27 utilizando las frecuencias de los clústeres. Como resultado se presenta la tabla 31. A continuación se presenta la forma como se calcula dicho valor entre el centroide 0 y o0. Se calculan por separado los tipos de atributo. Para los atributos categóricos se utiliza la frecuencia, por ejemplo para el valor de o0 que es “Casado” y su valor es Si, en clúster 0 este valor en ese atributo tiene una frecuencia de 0.667, y se sustituye en 27, repitiendo los pasos los demás atributos de este tipo.

$d(o0, c0) = ((0 – 0.667)^2 + (3 – 2.667)^2)^{½} + ((1 – 0.667) +(1 – 0.333) + (1 - 0.667))$
$d(o0, c0) = (0.445 + 0.111)^{½} + (0.333 + 0.667 +0.333)$
$d(o0, c0) = (0.746) + (1.333)$
$d(o0, c0) = 2.079$

De la misma forma como se calculan las demás distancias entre representantes y objetos obteniendo como resultado lo presentado en la tabla 31.

Tabla 31: Distancia entre los objetos y los clústeres.

Objeto	Distancia clúster 0	Distancia clúster 1
o0	2.079	3.937
o1	4.093	2.202
o2	3.134	2.0
o3	3.700	2.387
o4	2.138	3.610
o5	1.745	2.869

Paso 6: como no hubo cambios, el algoritmo llego a un punto de convergencia. Los clústeres formados son serán clúster 0 = {o0, o4, o5} y el clúster 1 = {o1, o2, o3} y se presentan en la tabla 32.

Tabla 32: Resultado final del algoritmo k-frequency en empleados con k = 2.

Objeto	Casado	Coche	Hijos	Antigüedad	Sexo	Clúster
o0	Sí	No	0	3	H	0
o1	Sí	No	3	2	M	1
o2	Si	Si	2	1	H	1
o3	No	No	1	1	M	1
o4	No	Sí	1	3	M	0
o5	Si	Si	1	2	H	0

3.2 CLUSTERING JERÁRQUICO AGLOMERATIVO

Un método jerárquico crea una descomposición jerárquica de un conjunto de datos, formando un dendograma (árbol) que divide recursivamente el conjunto de datos en conjuntos cada vez más pequeños o más grandes. Teniendo en cuenta la gran funcionalidad de este tipo de algoritmos en aspectos como la clasificación u organización de conjunto de datos Rasemus implementa los pasos básicos de clustering jerárquico de tipo aglomerativo (ver figura 28), el cual es el más utilizado teniendo en cuenta tres tipos de enlace entre clústeres, enlace mínimo o single link, máxima o complete link y enlace promedio o averange también conocido como UPMGA.

En el clustering ascendente jerárquico también conocido como bottom - up se pretende ir agrupando en cada paso aquellos 2 objetos (o conglomerados) más cercanos, para de esta forma ir construyendo una estructura conocida como dendograma, la cual parte en su base de tantos conglomerados como objetos a

clasificar, los cuales son agrupados naturalmente en un único grupo conteniendo todos los objetos.

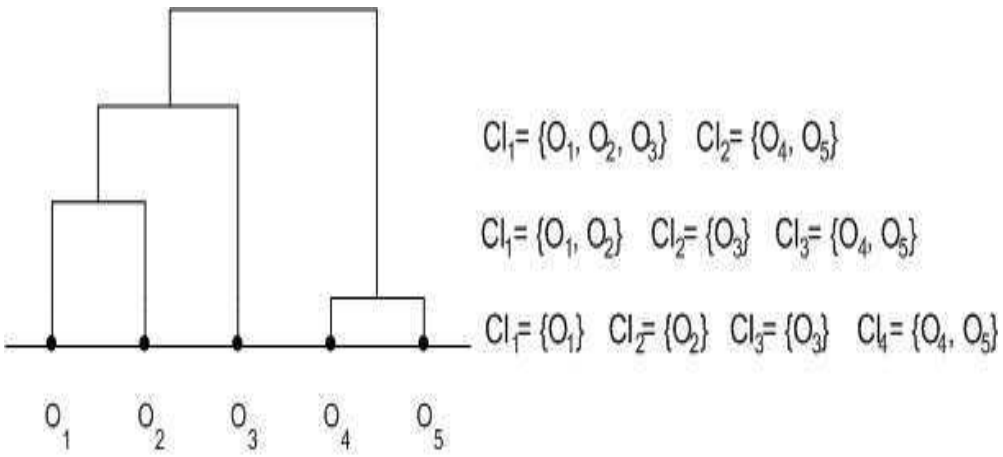
Figura 28: Pasos básicos algoritmo jerárquico aglomerativo.

Pasos Básicos Clustering Jerárquico Aglomerativo

1. Hacer cada objeto en un conglomerado (Cluster)
2. **Repita**:
   3. Medir distancia entre conglomerados
   4. Unir conglomerados con menor distancia
5. **Hasta** cuando solo haya un conglomerado o cuando se determine un número de iteraciones o número de niveles.

Si bien el costo computacional asociado a un clustering ascendente jerárquico es superior al que se relaciona con un clustering particional, el dendograma que se obtiene con él es más rico que una simple partición, ya que posibilita la obtención de distintas particiones, simplemente variando el nivel de corte de dicha estructura, tal y como se observa en la figura 29. De esta forma la problemática a la que se aludía en el apartado del clustering particional relativa a la determinación a priori del valor de k (número de grupos) queda solventada.

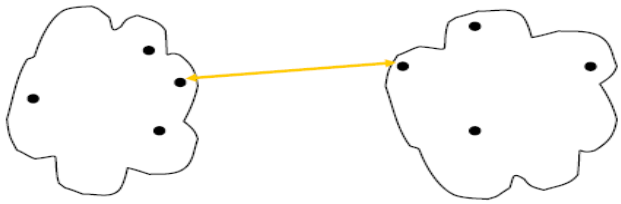
Figura 29: Dendograma resultado de la clasificación ascendente jerárquica



3.2.1 Funciones para medir distancia entre conglomerados.

Enlace mínimo o single–link: Considera la distancia más pequeña entre los elementos de los dos grupos. Es decir calcula la distancia de cada punto del clúster C1 con el C2 y la menor distancia se toma, como se muestra en la figura 30.

Figura 30: Representación enlace mínimo.



$$d_{min}(C_i, C_j) = min_{\forall p \in C_i, \forall q \in C_j} \|p - q\|$$

A continuación se presentan algunas características:

- ✓ No es útil para resumir datos.
- ✓ Útil para detectar outliers (estarán entre los últimos en unirse a la jerarquía).
- ✓ Pueden usarse medidas de la similitud o de la disimilitud.
- ✓ Tiende a construir clústeres demasiado grandes y sin sentido.
- ✓ Invariante bajo transformaciones monótonas de la matriz de distancias.

Ejemplo 5: algoritmo clustering jerárquico con el conjunto de datos empleados utilizando enlace mínimo.

Se trata de obtener un dendograma correspondiente a los 6 objetos o0, o1, o2, o3, o4, o5 pertenecientes al conjunto de datos empleados (ver tabla 1). A demás de la distancia entre grupos para los enlaces, se necesita la distancia entre objetos, para ello se utilizara la distancia euclidiana y la disimilitud de hamming para los atributos numéricos y categóricos.

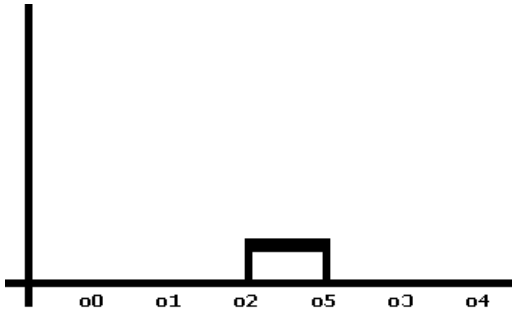
Paso 1: calcular la Matriz de distancia entre objetos (ver tabla 33). Solo se calculara una matriz triangular ya que la distancia entre d(oi,oj) = d(oj,oi).

Tabla 33: Matriz triangular de distancia entre objetos del conjunto de datos empleados.

	o0	**o1**	**o2**	**o3**	**o4**	**o5**
o0	0	4.162	3.828	4.236	4	2.414
o1		0	3.414	3.236	4.236	4
o2			0	4	4.236	1.414
o3				0	3	4
o4					0	3
o5						0

Paso 2: Hacer cada objeto como conglomerado, es decir para este caso tendíamos 6 clústeres. Y unimos los dos clústeres que tengan la menor distancia. Para este caso sería o2 y o5.El dendograma parcial se muestra en la figura 31.

Figura 31: Dendograma parcial, resultado de la primera agrupación en empleados con enlace-mínimo.



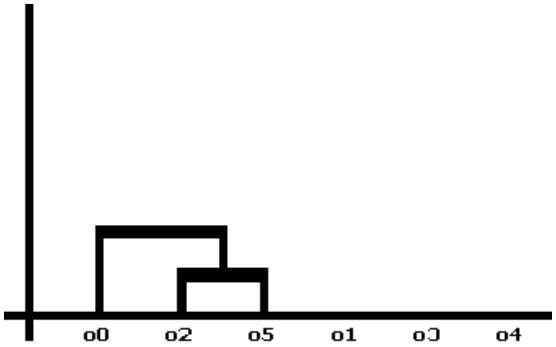
Paso 3: Se calcula nuevamente la distancia triangular entre clústeres, teniendo en cuenta que ya se unieron los objetos o2 y o5. Para determinar la distancia entre clústeres se aplicara el concepto de enlace mínimo anteriormente mencionado (ver tabla 34). Por ejemplo para el caso de la distancia entre o0 y {o2, o5} es 2.414 es el resultado de la distancia entre o0 y o5, ya que la distancia entre o0 y o2 es 3.828.

Tabla 34: Matriz de distancia entre clústeres con enlace mínimo después de la unión de o2 y o5.

	o0	**o1**	**{o2,o5}**	**o3**	**o4**
o0	0	4.162	2.414	4.236	4
o1		0	3.414	3.236	4.236
{o2,o5}			0	4	3
o3				0	3
o4					0

Paso 4: se unen los clústeres con menor distancia teniendo en cuenta la tabla 34. En ese caso sería o1 con {o2,o5}. El dendograma parcial se muestra en la figura 32.

Figura 32: Dendograma parcial después de la unión entre o0 y {o2, o5},



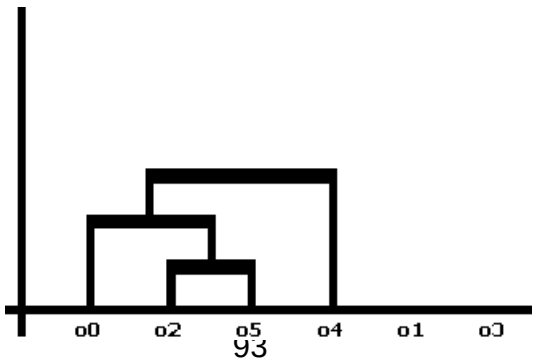
Paso 5: Se calcula nuevamente la distancia triangular entre clústeres, teniendo en cuenta que ya se unieron los clústeres o0 y {o2, o5} (ver tabla 35).

Tabla 35: Matriz de distancia entre clústeres con enlace mínimo después de la unión de o0 y {o2, o5}.

	{o0,o2,o5}	**o1**	**o3**	**o4**
{o0,o2,o5}	0	3.414	4	3
o1		0	3.236	4.236
o3			0	3
o4				0

Paso 6: se unen los clústeres con menor distancia teniendo en cuenta la tabla 35. En ese caso sería o4 con {o0, o2, o5}. El dendograma parcial se muestra en la figura 33.

Figura 33: Dendograma parcial después de la unión entre o4 y {o0, o2, o5},



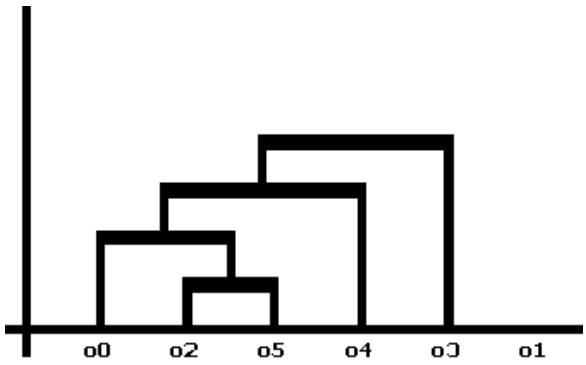
Paso 7: Se calcula nuevamente la distancia triangular entre clústeres, teniendo en cuenta que ya se unieron los clústeres o4 y {o0, o2, o5} (ver tabla 36).

Tabla 36: Matriz de distancia entre clústeres con enlace mínimo después de la unión de 04 y {o0, o2, o5}.

	{o0,o2,o5,o4}	**o1**	**o3**
{o0,o2,o5,o4}	0	3.414	3
o1		0	3.236
o3			0

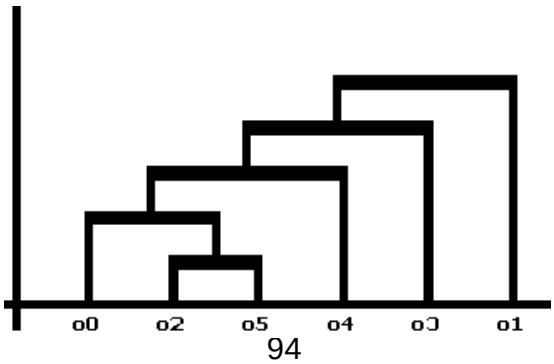
Paso 8: se unen los clústeres con menor distancia teniendo en cuenta la tabla 36. En ese caso sería o3 con {o0, o2, o5}. El dendograma parcial se muestra en la figura 34.

Figura 34: Dendograma parcial después de la unión entre {o0, o2, o5, o4} y o3.



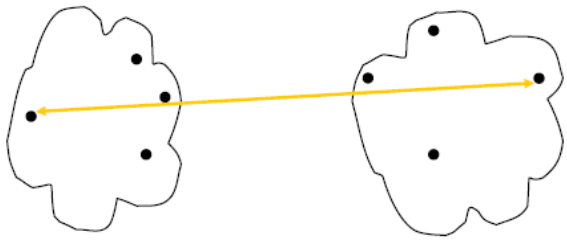
Paso 9: finalmente solo quedan los clústeres {o0, o2, o5, o4. o3} y o1, estos dos clústeres se unen y obtenemos el dendograma final presentado en la figura 35.

Figura 35: Dendograma final después de aplicar el algoritmo de clustering aglomerativo con enlace mínimo en empleados.



Enlace máximo o complete–link: Considera la distancia más grande entre los elementos de los grupos. Es decir calcula la distancia de cada punto del clúster C1 con el C2 y la mayor distancia se toma. Como se muestra en la figura 36.

Figura 36: Representación enlace máximo.



$$d_{max}(C_i, C_j) = max_{\forall p \in C_i, \forall q \in C_j} \|p - q\|$$

Algunas características se presentan a continuación.

- ✓ Útil para detectar outliers.
- ✓ Pueden usarse medidas de la similitud o de la disimilitud.
- ✓ Tiende a construir clústeres pequeños y compactos.
- ✓ Invariante bajo transformaciones monótonas de la matriz de distancias.

Ejemplo 6: algoritmo clustering jerárquico con el conjunto de datos empleados utilizando enlace máximo.

Se trata de obtener un dendograma correspondiente a los 6 objetos o0, o1, o2, o3, o4, o5 pertenecientes al conjunto de datos empleados (ver tabla 1). A demás de la distancia entre grupos para los enlaces, se necesita la distancia entre objetos, para ello se utilizara la distancia euclidiana y la disimilitud de hamming para los atributos numéricos y categóricos.

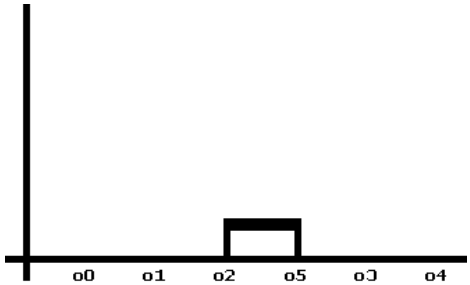
Paso 1: calcular la Matriz de distancia entre objetos (ver tabla 37). Solo se calculara una matriz triangular ya que la distancia entre d(oi,oj) = d(oj,oi).

Tabla 37: Matriz triangular de distancia entre objetos del conjunto de datos empleados.

	o0	**o1**	**o2**	**o3**	**o4**	**o5**
o0	0	4.162	3.828	4.236	4	2.414
o1		0	3.414	3.236	4.236	4
o2			0	4	4.236	1.414
o3				0	3	4
o4					0	3
o5						0

Paso 2: Hacer cada objeto como conglomerado, es decir para este caso tendíamos 6 clústeres. Y unimos los dos clústeres que tengan la menor distancia. Para este caso sería o2 y o5 (ver tabla 37).El dendograma parcial se muestra en la figura 37.

Figura 37: Dendograma parcial, resultado de la primera agrupación en empleados con enlace-máximo.



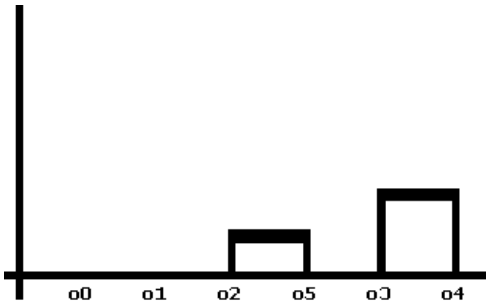
Paso 3: Se calcula nuevamente la distancia triangular entre clústeres, teniendo en cuenta que ya se unieron los objetos o2 y o5. Para determinar la distancia entre clústeres se aplicara el concepto de enlace máxima anteriormente mencionado (ver tabla 38). Por ejemplo para el caso de la distancia entre o0 y {o2, o5} es 3.828 es el resultado de la distancia entre o0 y o2, ya que la distancia entre o0 y o5 es 2.414.

Tabla 38: Matriz de distancia entre clústeres con enlace máximo después de la unión de o2 y o5.

	o0	**o1**	**{o2,o5}**	**o3**	**o4**
o0	0	4.162	3.828	4.236	4
o1		0	4	3.236	4.236
{o2,o5}			0	4	4.236
o3				0	3
o4					0

Paso 4: se unen los clústeres con menor distancia teniendo en cuenta la tabla 38. En ese caso sería o3 y o4. El dendograma parcial se muestra en la figura 38.

Figura 38: Dendograma parcial después de la unión entre o3 y o4,



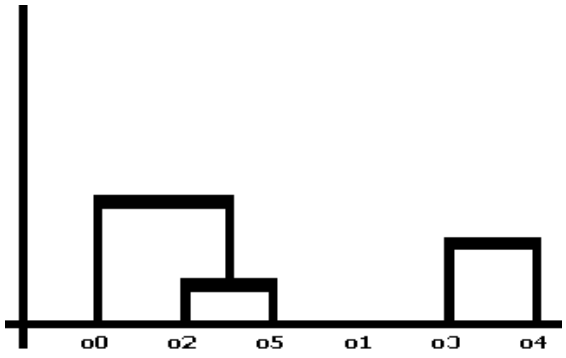
Paso 5: Se calcula nuevamente la distancia máxima triangular entre clústeres, teniendo en cuenta que ya se unieron los clústeres {o3.o4} (ver tabla 39).

Tabla 39: Matriz de distancia entre clústeres con enlace máximo después de la unión de o3 y o4.

	o0	**o1**	**{o2,o5}**	**{o3,o4}**
o0	0	4.162	3.828	4.236
o1		0	4	4.236
{o2,o5}			0	4.236
{o3,o4}				0

Paso 6: se unen los clústeres con menor distancia teniendo en cuenta la tabla 39. En ese caso sería o0 con {o2, o5}. El dendograma parcial se muestra en la figura 39.

Figura 39: Dendograma parcial después de la unión entre o0 y {o2, o5},



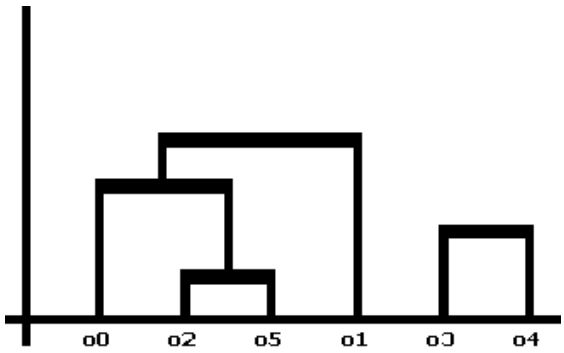
Paso 7: Se calcula nuevamente la distancia triangular entre clústeres, teniendo en cuenta que ya se unieron los clústeres o0 y {o2, o5} (ver tabla 40).

Tabla 40: Matriz de distancia entre los después de la unión de o0 y {o2, o5}.

	{o0,o2,o5 }	**o1**	**{o3,o4}**
{o0,o2,o5 }	0	4.162	4.236
o1		0	4.236
{o3,o4}			0

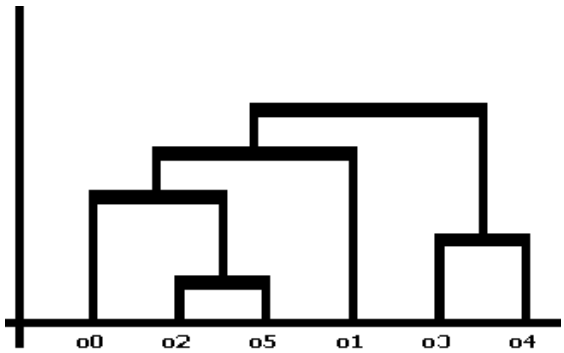
Paso 8: se unen los clústeres con menor distancia teniendo en cuenta la tabla 40. En ese caso sería {o0, 02, o5} y o1. El dendograma parcial se muestra en la figura 40.

Figura 40: Dendograma parcial después de la unión entre {o0, o2, o5 } y o1.



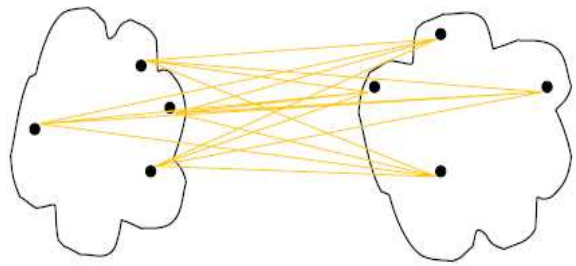
Paso 9: finalmente solo quedan los clústeres {o0, o2, o5, o1} y {o3, o4}, estos dos clústeres se unen y se obtiene el dendograma final presentado en la figura 41.

Figura 41: Dendograma final después de aplicar el algoritmo de clustering aglomerativo con enlace máximo en empleados.



Enlace promedio o averange–link: También conocido como algoritmo UPGMA. (Unweighted Pair-Groups Method Average). Considera la distancia promedio entre todos los elementos de los grupos.

Figura 42: Representación enlace promedio.



$$d_{avg}(C_i, C_j) = \frac{1}{|c_i||c_j|} \sum_{p \in C_i, q \in C_j} \|p - q\|$$

Algunas características se presentan a continuación.

- ✓ Proporciona clústeres ni demasiado grandes ni demasiado pequeños.
- ✓ Pueden utilizarse medidas de la similitud o de la disimilitud.
- ✓ No es invariante por transformaciones monótonas de las distancias.
- ✓ Tiende a fusionar clústeres con varianzas pequeñas y tiende a proporcionar clústeres con la misma varianza.
- ✓ Buena representación gráfica de los resultados.

Ejemplo7: algoritmo clustering jerárquico con el conjunto de datos empleados utilizando enlace promedio.

Se trata de obtener un dendograma correspondiente a los 6 objetos o0, o1, o2, o3, o4, o5 pertenecientes al conjunto de datos empleados (ver tabla 1). A demás de la distancia entre grupos para los enlaces, se necesita la distancia entre objetos, para ello se utilizara la distancia euclidiana y la disimilitud de hamming para los atributos numéricos y categóricos.

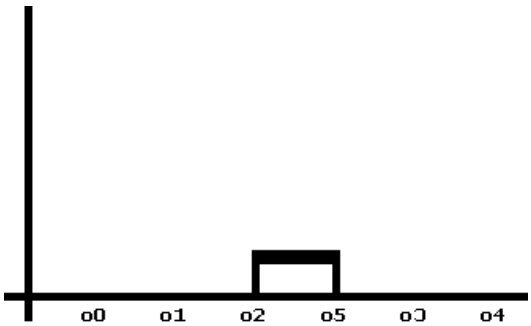
Paso 1: calcular la Matriz de distancia entre objetos (ver tabla 41). Solo se calculara una matriz triangular ya que la distancia entre d(oi,oj) = d(oj,oi).

Tabla 41: Matriz triangular de distancia entre objetos del conjunto de datos empleados.

	o0	**o1**	**o2**	**o3**	**o4**	**o5**
o0	0	4.162	3.828	4.236	4	2.414
o1		0	3.414	3.236	4.236	4
o2			0	4	4.236	1.414
o3				0	3	4
o4					0	3
o5						0

Paso 2: Hacer cada objeto como conglomerado, es decir para este caso tendíamos 6 clústeres. Y unimos los dos clústeres que tengan la menor distancia. Para este caso sería o2 y o5.El dendograma parcial se muestra en la figura 43.

Figura 43: Dendograma parcial, resultado de la primera agrupación en empleados con enlace-promedio.



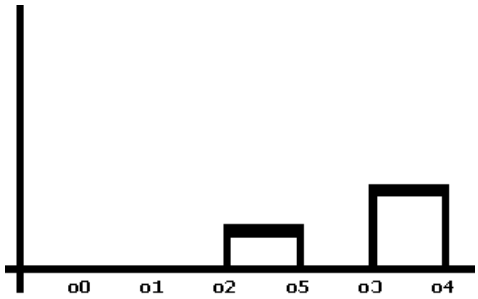
Paso 3: Se calcula nuevamente la distancia triangular entre clústeres, teniendo en cuenta que ya se unieron los objetos o2 y o5. Para determinar la distancia entre clústeres se aplicara el concepto de enlace promedio anteriormente mencionado (ver tabla 42). Por ejemplo para el caso de la distancia entre o0 y {o2, o5} es 3.121 es el resultado de la su distancia entre o0 y o2, y o0 y o5 dividida entre 2, que son el número de elementos.

Tabla 42: Matriz de distancia entre clústeres con enlace máximo después de la unión de o2 y o5.

	o0	**o1**	**{o2,o5}**	**o3**	**o4**
o0	0	4.162	3.121	4.236	4
o1		0	3.707	3.236	4.236
{o2,o5}			0	4	3.618
o3				0	3
o4					0

Paso 4: se unen los clústeres con menor distancia teniendo en cuenta la tabla 42. En ese caso sería o3 y o4. El dendograma parcial se muestra en la figura 44.

Figura 44: Dendograma parcial después de la unión entre o3 y o4,



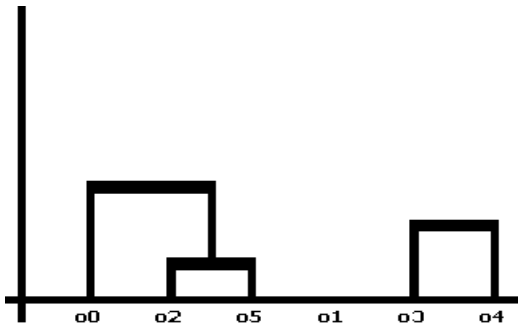
Paso 5: Se calcula nuevamente la distancia promedio triangular entre clústeres, teniendo en cuenta que ya se unieron los clústeres {o3.o4} (ver tabla 43).

Tabla 43: Matriz de distancia entre clústeres con enlace promedio después de la unión de o3 y o4.

	o0	**o1**	**{o2,o5}**	**{o3,o4}**
o0	0	4.162	3.121	4.118
o1		0	3.707	3.736
{o2,o5}			0	3.809
{o3,o4}				0

Paso 6: se unen los clústeres con menor distancia teniendo en cuenta la tabla 43. En ese caso sería o0 con {o2, o5}. El dendograma parcial se muestra en la figura 45.

Figura 45: Dendograma parcial después de la unión entre o0 y {o2, o5},



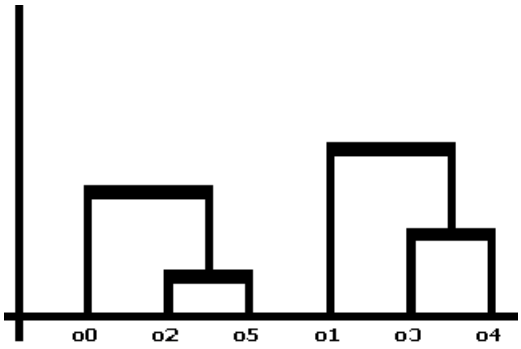
Paso 7: Se calcula nuevamente la distancia promedio triangular entre clústeres, teniendo en cuenta que ya se unieron los clústeres o0 y {o2, o5} (ver tabla 44).

Tabla 44: Matriz de distancia entre los después de la unión de o0 y {o2, o5}.

	{o0,o2,o5 }	**o1**	**{o3,o4}**
{o0,o2,o5 }	0	3.859	3.912
o1		0	3.736
{o3,o4}			0

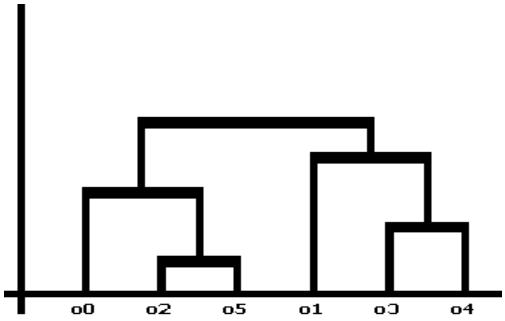
Paso 8: se unen los clústeres con menor distancia teniendo en cuenta la tabla 44. En ese caso sería o1 y {03, o4}. El dendograma parcial se muestra en la figura 46.

Figura 46: Dendograma parcial después de la unión entre o1 y {o3, o4}.



Paso 9: finalmente solo quedan los clústeres {o0, o2, o5 } y {o1, o3, o4}, estos dos clústeres se unen y obtenemos el dendograma final presentado en la figura 47.

Figura 47: Dendograma final después de aplicar el algoritmo de clustering aglomerativo con enlace promedio en empleados.



3.3 CLUSTERING BASADO EN DENSIDAD

Las técnicas basadas en densidad los clústeres se identifican por mirar la densidad entre sus puntos. Regiones con una alta densidad de puntos

representan la existencia de clústeres y las regiones con baja densidad de puntos indican el ruido de los clústeres o grupos de valores atípicos. Rasemus implementa los pasos básicos de la agrupación espacial basada en densidad aplicada con el ruido, más conocido como DBSCAN el cual ha tenido una gran aplicación en descubrimiento de densidades de datos de todo sentido.

3.3.1 Algoritmo DBSCAN

(Density Based Spatial Clustering of Applications with Noise). El algoritmo DBSCAN se introdujo por primera vez por Ester, et al. [22], y se fundamenta en un concepto basado en la densidad de clústeres. DBSCAN es un método basado en el supuesto de que la densidad de cada grupo, es mayor que los puntos de afuera; La densidad en la zona de ruidos es inferior a la densidad en cualquiera de los clústeres [22].

Este algoritmo es especialmente adecuado para hacer frente a grandes conjuntos de datos, con el ruido, y es capaz de identificar los grupos con diferentes tamaños y formas. Para ello este algoritmo necesita conocer dos parámetros. El primero es el valor conocido como épsilon, que es la medida de densidad que va tener en cuenta para crear un área de barrido y adjuntar los puntos al clúster formado. Otro parámetro necesario es el número mínimo de puntos encontrados en un clúster formado. Es decir si el número de puntos de un clúster es inferior a el mínimo de puntos requeridos el grupo no se tiene en cuenta. En el algoritmo DBSCAN se deben tener en cuenta tener en cuenta los siguientes conceptos [69].

Figura 48: Pasos algoritmo DBScan

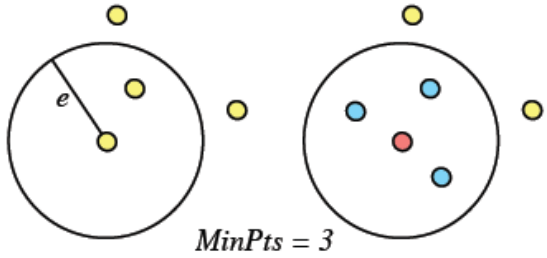
Pasos Básicos Algoritmo DBSCAN

1. Establecer los valores de épsilon y minPoints.
2. Tomar todos los registros como puntos no asignados
3. **Repita**:
   - 3.1 Asignar aleatoriamente un punto no asignado como core – point o nucleo
   - 3.2 **Repita:**
     - 3.2.1 Medir la distancia entre core – point y lo puntos no asignados
       - Si la distancia entre punto y core – point es menor o igual a épsilon
         - Entonces: se asigna el punto al core – point
   - 3.3 **Hasta:** Cuando ya no haya más puntos que medir
   - 3.4 Si la cantidad de puntos asignados es mayor o igual a minPoints
     - Entonces: establecer core – pint y sus puntos asignados como cluster
     - Sino: eliminar el core – point y restablecer los puntos su valor de no asignados
   - 3.5 **Hasta:** Cuando ya no haya más no asignados o cuando s deternine un numero de iteraciones.
4. **Hasta** Cuando solo haya un conglomerado o cuando se determine un número de iteraciones o número de niveles.

Core-point:

Para cualquier punto p en D, p es un punto núcleo si hay puntos dentro de su *e* – *neighborhood* o área de barrido y la cantidad de puntos encontrados es mayor o igual a una cantidad mínima de puntos determinada. Ver figura 49 el punto central.

Figura 49: Muestra del área de barrido de un punto



E - Neighborhood of a point.

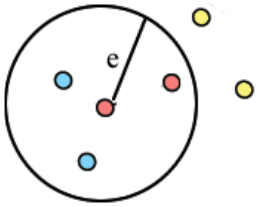
También se conoce como área de barrido. Denotado por $N_{€}(P)$, está definido, $N_{€}(P) = \{ q € D | dis(p,q) \leq e | \}$, donde D es el conjunto de datos dado, p € D, y dist(p,q) es una función de distancia que calcula la diferencia entre p y q. ver figura 49, donde épsilon es la línea que sale del punto central y el área está delimitada por el circulo.

El algoritmo DBSCAN tiene dos formas de generar los clústeres según la forma de determinar la forma de establecer las áreas de barrido y su forma. Las cuales se mencionan a continuación.

Density-reachable

Conocida también como densidad directa o accesible, que se consigue a través de un núcleo y la medición entre los puntos formando un clúster con los puntos que se encuentren dentro de la distancia épsilon, formando generalmente formas circulares, como se muestra en la figura 50.

Figura 50: density – reachable o densidad directa



Ejemplo 8: algoritmo DBScan con el conjunto de datos empleados utilizando distancia directa.

Se trata de obtener un áreas de los 6 objetos o0, o1, o2, o3, o4, o5 pertenecientes al conjunto de datos empleados (ver tabla 1). Aquí se utilizara la formación de áreas con el concepto de distancia directa.

Paso 1: establecer el valor de épsilon y el valor mínimo de puntos por núcleo. Para este caso utilizaremos un valor mínimo de punto de 1 y épsilon de 3.0.

Paso 2: Se toma un elemento de forma arbitraria en los objetos {o0, o1, o2, o3, o4, o5}, para el primer caso evaluaremos el o0 como core-point y se calculan distancias entre o0, y los objetos {o1, o2, o3, o4, o5}(ver tabla 45).

Tabla 45: Distancias entre core-point o0 y {o1, o2, o3, o4, o5}

	o1	**o2**	**o3**	**o4**	**o5**
o0	4.162	3.828	4.236	4	2.414

Paso 3: Evaluar el valor de épsilon con las distancias calculadas con o0 (ver tabla 45), tomando los valores inferiores o iguales al en este caso 3.0, y si el numero de objetos tomados es mayor o igual a 1 que es el número mínimo de puntos por core-point, formar un clúster con dichos puntos y eliminarlos del conjunto. Para el caso de o0 se encontró el objeto o5, Entonces como se cumple los requerimientos se forma el cluster 0 = {o0, o5} y ya no se los toma en cuenta.

Paso 4: Se toma un elemento de forma arbitraria en los objetos { o1, o2, o3, o4}, para en este caso evaluaremos el o2 como core-point y se calculan distancias entre o2, y los objetos {o1, o3, o4 }. Obteniendo el resultado presentado en la tabla 46.

Tabla 46: Distancias entre core-point o2 y {o1, o3, o4}

	o1	**o3**	**o4**
o2	3.414	4	4.236

Paso 5: Evaluar el valor de épsilon con las distancias calculadas con o2 (ver tabla 46) y el número mínimo de puntos por core-point. Para el caso de o2 no se cumplen los requerimientos, entonces se toma a o2 como ruido y no se agrupa.

Paso 6: Se toma un elemento de forma arbitraria en los objetos {o1, o3, o4}, para en este caso se evaluará el o1 como core-point y se calculan distancias entre o1, y los objetos {o3, o4}. Obteniendo el resultado presentado en la tabla 47.

Tabla 47: Distancias entre core-point o1 y {o3, o4}

	o3	o4
o1	3.236	4.236

Paso 7: Evaluar el valor de épsilon con las distancias calculadas con o1 (ver tabla 47) y el número mínimo de puntos por core-point. Para el caso de o1 no se cumplen los requerimientos, entonces se toma a o1 como ruido y no se agrupa.

Paso 8: Se toma un elemento de forma arbitraria en los objetos {o3, o4}, para en este caso se evaluara el o3 como core-point y se calculan distancias entre o3 y o4 que es 3.0. como o3 cumple con los requerimientos de épsilon y de mínimos puntos se forma el clúster 1 = {o3, o4}.

Paso 9: como ya no hay más objetos por evaluar el resultado final se presenta en la tabla 48.

Tabla 48: Resultado final del algoritmo DBScan con el conjunto de datos empleados definiendo épsilon = 3.0 y cantidad mínima de puntos = 1;

Objeto	Casado	Coche	Hijos	Antigüedad	Sexo	Clúster
o0	Sí	No	0	3	H	0
o1	Sí	No	3	2	M	Noise
o2	Si	Si	2	1	H	Noise
o3	No	No	1	1	M	1
o4	No	Sí	1	3	M	1
o5	Si	Si	1	2	H	0

Density-connected

Conocida también como densidad directa o accesible, que se consigue a través de un núcleo y la medición entre los puntos formando un clúster con los puntos que se encuentren dentro de la distancia épsilon y los puntos que son adicionados a dicho clúster, se establecen como core-point y crean una nueva área de barrido, adicionando los puntos que se encuentren ahí al clúster al cual fueron asignados. Este tipo de adición de áreas permite encontrar densidades de forma diferente. Como se muestra en la figura 51.

Figura 51: Density-connected o densidad conectada

Ejemplo 9: algoritmo DBScan con el conjunto de datos empleados utilizando distancia conectada.

Se trata de obtener un áreas de los 6 objetos o0, o1, o2, o3, o4, o5 pertenecientes al conjunto de datos empleados (ver tabla 1). Aquí se utilizará la formación de áreas con el concepto de distancia conectada.

Paso 1: establecer el valor de épsilon y el valor mínimo de puntos por núcleo. Para este caso utilizaremos un valor mínimo de punto de 1 y épsilon de 3.0.

Paso 2: Se toma un elemento de forma arbitraria en los objetos {o0, o1, o2, o3, o4, o5}, para el primer caso evaluaremos el o0 como core-point y se calculan distancias entre o0, y los objetos {o1, o2, o3, o4, o5} (ver tabla 49).

Tabla 49: Distancias entre core-point o0 y {o1, o2, o3, o4, o5}

	o1	**o2**	**o3**	**o4**	**o5**
o0	4.162	3.828	4.236	4	2.414

Paso 3: Evaluar el valor de épsilon con las distancias calculadas con o0 (ver tabla 49), tomando los valores inferiores o iguales al en este caso 3.0, y si el numero de objetos tomados es mayor eliminarlos del conjunto. Para el caso de o0 se encontró el objeto o5, Entonces como se cumple los requerimiento de épsilon entonces, como se adiciono o5, se debe calcular la distancia entre o5 y los objetos sobrantes (ver tabla 50).

Tabla 50: Distancias entre o5 y {o1, o2, o3, o4}

	o1	**o2**	**o3**	**o4**
o5	4	1.414	4	3

Paso 4: se toman los elementos de que cumplan que su distancia con o5 sea menor o igual que épsilon y se adicionan como puntos de o0, en este caso serian o2 el primer caso, se adiciona o2 a o0.

Paso 4.1: Se va a calcular la distancia entre o2 y los objetos restantes (ver tabla 51)

Tabla 51: Distancias entre core-point o2 y {o1, o3, o4}

	o1	**o3**	**o4**
o2	3.414	4	4.236

Paso 4.1: como ninguna distancia cumple la condición de épsilon. Se continúa evaluando con o5.

Paso 4.2: otro elemento que cumple con o5 el requerimiento de épsilon es o4. A o4 se adiciona como objeto de o0 y se calcula la distancia entre o4 y los objetos restantes (ver tabla 52).

Tabla 52: Distancias entre core-point o4 y {o1, o3 }

	o1	**o3**
o4	4.236	3

Paso 4.2: se toman los elementos de que cumplan que su distancia con o4 sea menor o igual que épsilon y se adicionan como puntos de o0 (ver tabla 52), en este caso serian o3, se adiciona o3 a o0.

Paso 4.2.1: Se va a calcular la distancia entre o3 y los objetos restantes, solo quedaría o1 y la distancia entre o3 y o1 es 3.236, que no es menor ni igual que épsilon.

Paso 5: como ya no hay mas elemento y se revisas si o0 cumple con los requerimientos de épsilon y de mínimos puntos, como se o0 adiciono a los o5, o2, o4 y o3, con el concepto de distancia conectada, cumple con los puntos mínimos y se forma el clúster 0 = {o0, o2, o3, o4, o5}.

Paso 6: se va a evaluar los objetos restantes que sería o1, pero como ya no hay más objetos por evaluar se da por finalizado. el resultado final se presenta en la tabla 53.

Tabla 53: Resultado final del algoritmo DBScan con el conjunto de datos empleados defiendo épsilon = 3.0 y cantidad mínima de puntos = 1 utilizando distancia conectada.

Objeto	**Casado**	**Coche**	**Hijos**	**Antigüedad**	**Sexo**	**Clúster**
o0	Sí	No	0	3	H	0
o1	Sí	No	3	2	M	Noise
o2	Si	Si	2	1	H	0
o3	No	No	1	1	M	0
o4	No	Sí	1	3	M	0
o5	Si	Si	1	2	H	0

4. ANALISIS DE HERRAMIENTAS DE CLUSTERING

En la actualidad existen varias grupos de desarrollo que el mundo que se han interesado en la minería de datos y han desarrollados herramientas que implementan diferentes algoritmos de extracción de conocimiento. A continuación se presentan algunas de ellas, las cuales tienen entre sus técnicas implementadas el clustering.

4.1 HERRAMIENTAS DE DISTRIBUCIÓN LIBRE

4.1.1 Adam

Desarrollada por el Centro de Tecnología de Información y Sistemas de la Universidad de Alabama en Huntsville. Adam (The Algorithm Development and Mining System) [75] es un kit de Herramientas de Minería de Datos diseñado para el uso científico y tratamiento de imágenes. Incluye modelos de reconocimiento, procesamiento de imágenes, optimización, y reglas de asociación. Adam no contiene análisis estadístico avanzado, conversión de formatos, no contiene herramientas para la visualización de conjunto de datos. El sistema consta de un conjunto de componentes individuales que pueden ser utilizados juntos para realizar tareas complejas. Los componentes son empaquetados como ejecutables y módulos de Python, que tienen todos los componentes como las funciones de módulo y pueden ser llamados a través de scripts.

No posee interfaz grafica, todos sus procedimientos deben ser llamados a través de líneas de comando. La fuente de datos debe estar almacenada en archivos planos, es decir no es posible conectarse con ninguna base de datos. Además todo este todos los resultados son entregados por medio de archivos, que para su interpretación de deben tener muy en claro los conceptos de minería de datos y sobre todo de clustering.

Sistemas Operativos: Todos**.**

Técnicas de clustering que contempla:

Algoritmos implementados:

ITSC_DBSCAN utilizando el algoritmo DBSCAN
ITSC_Isodata utilizando el algoritmo Isodata
ITSC_HierarchicalCluster utilizando agglomerative hierarchical clustering
ITSC_KMeans utilizando el algoritmo K-Means
ITSC_KMediods Cluster datos utilizando el algoritmo K-Mediods
ITSC_Maximin Cluster datos utilizando el algoritmo Maximin

4.1.2 AlphaMiner

Desarrollado por el Instituto tecnológico de E-Bussines de la universidad Hong Kong, bajo el apoyo del Fondo de la Innovación y Tecnología (ITF) del Gobierno de Hong Kong Regional Administrativa Especial (RAE). AlphaMiner [70] es una plataforma de minería de datos, desarrollado bajo tecnologías de código abierto, ofrece herramientas que permiten desarrollar todas las etapas del proceso KDD. En la etapa de minería contempla algoritmos de asociación, clasificación, regresión logística y clustering.

AlphaMiner se ajusta al proceso estándar de minería de datos, que está diseñado para hacer proyectos de minería de datos más rápido, más baratos, más fiables y más manejable, y define el proceso de minería de datos en seis etapas:

1. La comprensión de negocios
2. La comprensión de los datos
3. La preparación de datos
4. Modelado
5. La evaluación
6. El despliegue

AlphaMiner proporciona una interfaz Drag & Drop para la realización del flujo de trabajo, permitiendo a los usuarios construir el proceso de selección de las operaciones necesarias. Además, los usuarios pueden tener más de un flujo de proceso en cada proyecto de minería de datos. Y, para cada flujo de proceso, la entrada de datos, las transformaciones, modelos de construcción, modelos de evaluación y modelo despliegue se pone claramente de manifiesto.

El lenguaje de programación en el que fue desarrollado es Java™, lo que permite que la fuente de datos pueda ser escogida desde distintos gestores de bases de datos a través de la conectividad JDBC. También permite como fuentes de datos archivos de Excel, archivos con formatos arff, de texto y demás.

AlphaMiner es un componente que está basado en una plataforma, que controla la ejecución y el flujo de trabajo de todos componentes integrados. En un principio se había integrado con dos muy populares herramientas de minería de datos de

código abierto, Weka y Xelopes que proporcionan un amplio conjunto de funciones de minería de datos.

Sistema Operativo: Win

Algoritmos Implementados:
K-means.

4.1.3 CLUTO - Familia de Herramientas para la agrupación de datos

CLUTO es un paquete de software desarrollado por el Departamento de Ciencias de la Computación e Ingeniería en la Universidad de Minnesota en las Ciudades Gemelas de Minneapolis y Saint Paul y el Centro de Tecnología Digital (DTC) en la Universidad de Minnesota. Cluto [77] sirve para la aplicación de técnicas de clustering y para analizar las características de algunas de ellas. CLUTO está preparado para agrupar los conjuntos de datos se que se plantean en diversas áreas de aplicación incluyendo la recuperación de información, las transacciones de compra del cliente web, sistemas de información geográfica, la ciencia y la biología.
Se puede trabajar en tres presentaciones:

CLUTO (Software para Clustering para Datasets de alta dimensión)
Es un conjunto de bibliotecas que a través de un programa de aplicación puede acceder directamente a las distintos tipos de clustering y análisis de los mismos.
Entre sus características encontramos que hay implementados diferentes tipos de clustering como lo son las técnicas de clustering particional, aglomerativo y particionamiento grafico básico. También se puede encontrar diferentes tipos de cálculo de distancias, como la distancia euclidiana, coseno, coeficiente de correlación, extensión Jaccard y también puede ser definida por el usuario.

gCLUTO
Es un conjunto de bibliotecas para visualizar los resultados de la agrupación utilizando arboles, matrices y visualizaciones de tipo montaña utilizando la biblioteca grafica OpenGL

wCLUTO
Es una aplicación web habilitada para la agrupación de datos, está diseñada para el agrupamiento de datos y el análisis de las necesidades de datos de genes. wCLUTO hace uso de las bibliotecas Cluto. Los usuarios pueden subir sus bases de datos, seleccionar entre varios métodos de agrupación, realizar el análisis en el servidor, y visualizar los resultados finales. El servidor web WCLUTO es auspiciado por el Centro de Genómica Computacional y Bioinformática en la Universidad de Minnesota.

Las bibliotecas CLUTO son llamadas a través de línea de comandos, está desarrollada para personas que tengan alto conocimiento en técnicas de agrupación.

Sistema Operativo: Win y Linux

Algoritmos Implementados:

- ✓ Clustering Particional
- ✓ Clustering Aglometarivo
- ✓ Single - link
- ✓ Complete - link
- ✓ UPGMA

4.1.4 Databionic ESOM

Es un conjunto de programas para realizar tareas de minería de datos como clustering, visualización, clasificación y mapas emergentes de auto-organización (ESOM). desarrollados por Databionics Grupo de Investigación en la Universidad de Marburg, Alemania. Las herramientas ESOM están escritas en java para la máxima portabilidad, además de ser publicadas bajo términos de licencia GPL. Entre las características que se pueden encontrar, formación de ESOM con diferentes métodos de inicialización, implementación de algoritmos, funciones de distancia, topologías de red ESOM, y los núcleos de barrido, visualización de alta definición para datos espaciales con U-Matrix, Matrix-P, Componente Planos, SDH, análisis exploratorio de datos y la agrupación de ESOM vinculado a la formación de datos, clasificación de datos, descripciones.

Sistema Operativo: Win y Linux

Algoritmos Implementados

la agrupación de la ESOM puede realizarse en dos niveles diferentes. En primer lugar, la bestmatches y, por tanto, los datos correspondientes a los puntos pueden ser agrupados manualmente en varios grupos. No todos los puntos tienen que ser etiquetados, los valores suelen ser fácilmente detectados y se puede quitar. El grupo de miembros puede ser visualizado por una coloración de la bestmatches. En segundo lugar, las neuronas pueden ser agrupadas, de esta manera las regiones del mapa que representan a un grupo pueden ser identificadas y utilizadas para clasificación de nuevos datos. Las regiones pueden ser visualizadas utilizando semi-transparencias en el color de las regiones con el fin de no superponer las visualizaciones. La visualización puede mostrar si existe

realmente una estructura de grupo. Alejadas de los puntos de datos van a ser también fácilmente detectables.

4.1.5 Knime

Es una plataforma de de exploración de datos que permite al usuario crear visualmente los flujos de datos a través de su interfaz Drag & Drop, ejecuta selectivamente algunos o todos los pasos de análisis y, posteriormente, investigar los resultados interactivamente a través de visualizaciones sobre los datos y los modelos.

KNIME [88] se desarrolló (y seguirá siendo ampliada) en la Cátedra de Bioinformática y Minería de Información en la Universidad de Konstanz, Alemania. El grupo encabezado por Michael Berthold también está utilizando KNIME para la enseñanza y la investigación en la Universidad.

KNIME versión de la base ya incorpora más de 100 nodos de procesamiento de datos I/O, preprocesamiento y limpieza, modelización, análisis y minería de datos, así como diversos puntos de vista interactivos, como los gráficos de dispersión, las coordenadas paralelas y otros. Incluye todos los módulos de análisis de Weka y plugins adicionales permiten trabajar con R-scripts (www.r-project.org), ofrece acceso a una vasta biblioteca de rutinas estadísticas.

KNIME se basa en la plataforma Eclipse y, a través de su API modular y fácilmente extensible.

Sistema Operativo: Win y Linux

Algoritmos de Implementados:

- ✓ Fuzzy c-Means
- ✓ K- Means
- ✓ SOTA Learner
- ✓ SOTA Predictor

4.1.6 Orange

Orange [91] es un componente basado en software de minería de datos. Incluye una serie de pasos como el preprocesamiento, la elaboración de modelos de datos y técnicas de exploración. Se basa en componentes C + +, que son accesibles, ya sea directamente (no muy frecuente), a través de scripts de Python (más fácil y mejor), o a través de objetos GUI llamados por Orange.

Orange se distribuye gratuitamente bajo licencia GPL. Es desarrollado por el laboratorio AI de la Facultad de Informática y Ciencias de Información de la

Universidad de Ljubljana de Trzaska en Ljubljana, Eslovenia. Posee una interfaz Drag & Drop y algunas características son:

- ✓ Los datos de entrada / salida: Orange puede leer y escribir archivos separados por tabuladores o archivos C4.5 y otros, y apoya también algunos formatos más exóticos.

- ✓ Preprocesamiento: se pueden creación de subconjuntos, discretización, característica de estimación de utilidad para las tareas de predicción.

- ✓ Elaboración de modelos de predicción: clasificación de árboles, clasificación Bayesina, k-NN, la mayoría clasificadores, máquinas de soporte de vectores, regresión lógica, basado en normas de clasificación (por ejemplo, CN2).

- ✓ Métodos para la descripción de datos: varios visualizadores, auto-organización de mapas, agrupación jerárquica, agrupación con el algoritmo k-means, escala multi-dimensional, y otros.

- ✓ Modelo de técnicas de validación, que incluyen diferentes datos de muestreo y validación de técnicas (como la validación cruzada, un muestreo aleatorio, etc), y diversas estadísticas para la validación de modelos (clasificación exactitud, AUC, la sensibilidad, especificidad...).

Sistema Operativo: Win y Linux

Algoritmos Implementados:

- ✓ K-means
- ✓ Hierarchical Clustering

4.1.7 PermutMatrix

PermutMatrix [95], Software para el análisis y visualización de datos (*Clustering and seriation analysis*) es un programa gráfico para realizar el análisis de un conjunto de datos. Desarrollada por G. Caraux y S. Pinloche integrantes del laboratorio de informática, robotica y microelectronica de Montpellier **(LIRMM)**, Francia. Esta Herramienta lleva a cabo varios tipos de análisis de agrupamiento jerárquico y una variedad de métodos estadísticos para la serialización.

Los datos numéricos se convierten en una gradación de colores. Las filas y/o columnas son permutadas de tal manera que parecen una estructura. Una fila o columna se puede ordenar de tal forma que puedan revelar una organización estadística en el conjunto de datos.

Este enfoque ha sido ampliamente utilizado durante muchos años en la arqueología, la sociología y la investigación operativa. Dispone de nuevas aplicaciones en biología, debido a la utilización de agrupación jerárquica en datos con información de genes.

Sistema Operativo: Win

Algoritmos Implementados:

Hierarchical clustering

- ✓ Complete linkage
- ✓ Single linkage
- ✓ McQuitty ‘s method (WPGMA)
- ✓ Average linkage (UPGMA)
- ✓ Ward’s minimum variance

4.1.8 Keel

Keel [87] es una herramienta de software para evaluar algoritmos evolutivos para problemas de minería de datos entre ellos se encuentran la regresión, clasificación, clustering, búsqueda de patrones de la minería y así sucesivamente. Contiene una gran colección de clásicos algoritmos de extracción de conocimiento, técnicas de preprocesamiento (ejemplo de selección, función de selección, discretización, métodos de imputación para valores, etc), Inteligencia Computacional de aprendizaje basado en algoritmos evolutivos incluyendo reglas de aprendizaje de algoritmos basados en diferentes enfoques (Pittsburgh, Michigan y la IRL,...), y modelos híbridos como sistemas difusos genéticos, redes neuronales evolutivas, etc. Nos permite realizar un análisis completo de cualquier modelo de aprendizaje en comparación con los existentes, incluye un módulo de prueba estadística para la comparación. Por otra parte, Keel se ha diseñado con un doble objetivo: la investigación y la educación.

Sistema Operativo: Win y Linux

Algoritmos de Utilizados:

- ✓ Kmeans

4.1.9 RapidMiner

RapidMiner [96], anterior YALE, desarrollado por la compañía Rapid-I de Dortmund, Alemania. Se convierte en una de las aplicaciones más utilizadas de minería de datos y soluciones de análisis predictivo en todo el mundo.

RapidMiner es en realidad una aplicación de minería de datos y un motor de inteligencia de negocios que también abarca muchos aspectos relacionados con el alcance de ETL (Extract, Transform y Load) a lo largo de Análisis para la presentación de informes. Tiene acceso a diferentes tipos de datos, como formatos ARFF, archivos planos delimitados por caracteres y también archivos de EXCEL, tiene una interfaz muy manejable ya que si el usuario no tiene experiencia realiza proyectos a través de Wizard, que va guiando paso a paso al usuario en el experimento. Además está provista de un sin número de algoritmos que abarcan todas las etapas de descubrimiento de conocimiento. Además entre sus algoritmos se pueden encontrar los algoritmos de Weka [100]. Es una de las herramientas más completas que se pueden encontrar en la actualidad. Esta desarrollada en Java.

Sistema Operativo: Todos

Algoritmos Implementados:

- ✓ AgglomerativeClustering
- ✓ AgglomerativeFlatClustering
- ✓ BregmanHardClustering
- ✓ ClusterModel2ExampleSet
- ✓ DBScanClustering
- ✓ ExampleSet2ClusterConstraintLis
- ✓ ExampleSet2ClusterModel
- ✓ FlattenClusterModel
- ✓ KMeans
- ✓ KMedoids
- ✓ KernelKMeans
- ✓ MPCKMeans
- ✓ RandomFlatClustering
- ✓ SupportVectorClustering
- ✓ TopDownClustering
- ✓ TopDownRandomClustering
- ✓ UPGMAClustering
- ✓ Weka
  - W-CLOPE
  - W-Cobweb
  - W-EM
  - W-FarthestFirst
  - W-SimpleKMeans
  - W-XMeans
  - W-sIB

4.1.10 PSPP

PSPP [78] originalmente llamado “Fiasco”, es una herramienta para el análisis estadístico de datos. analiza los datos, y descripción de los resultados. Puede realizar estadísticas descriptivas, pruebas t, regresión lineal y pruebas no paramétricas. Puede utilizar PSPP con su interfaz gráfica o la sintaxis de los comandos más tradicionales igual a SPSS.

Sistema Operativo: Todos.

Algoritmos de clustering implementados:

- ✓ Clustering Jerárquico
- ✓ TwoStep Cluster
- ✓ Quick Cluster.

4.1.11 Tanagra

Tanagra es software de minería de datos para el sector académico y de investigación. Desarrollada por Ricco RAKOTOMALALA en la ciudad de Lyon, Francia. Se proponen varios métodos de minería de datos, análisis exploratorio de datos, modelos estadísticos de aprendizaje, aprendizaje automático y bases de datos.

Este proyecto es el sucesor de SIPINA que implementa diversos algoritmos de aprendizaje supervisado, especialmente interactivo y manejando un entorno visual para la construcción de árboles de decisión. Tanagra contiene algunas técnicas de aprendizaje supervisado, paradigmas como el clustering, análisis factorial, estadísticas paramétricas y no paramétricas, reglas de asociación, característica de selección y construcción de algoritmos.

Tanagra es un “proyecto de código abierto”, ya que cada investigador puede tener acceso al código fuente, y añadir sus propios algoritmos, por lo que está de acuerdo y se ajusta a la licencia de distribución de software. El objetivo principal de Tanagra proyecto es dar a los investigadores y estudiantes un software de minería de datos fácil de utilizar, conforme a las presentes normas de desarrollo de software en este ámbito (especialmente en el diseño de su interfaz gráfico y la forma de utilizarlo), y que permite analizar, ya sea datos reales o de prueba.

El segundo propósito de Tanagra es la de proponer a los investigadores una arquitectura que les permite añadir fácilmente sus propios métodos de minería de datos, para comparar sus interpretaciones o ejecuciones. Tanagra actúa más

como una plataforma experimental con el fin de dejarlos ir a lo esencial de su trabajo, prescindir de ellos para hacer frente a la parte desagradable en la programación de este tipo de herramientas: la gestión de datos.

Sistema Operativo: Win

Algoritmos de clustering implementados:

- ✓ K-means
- ✓ Kohonen's Self Organization Map
- ✓ Kohonen's Learning Vector Quantizers
- ✓ Hierachical agglomerative clustering

4.1.12 Weka.

Weka [100] (Waikato Environment for Knowledge Analysis - *Entorno para Análisis del Conocimiento de la Universidad de Waikato*). es una colección de algoritmos de aprendizaje automático para tareas de minería de datos. Los algoritmos pueden ser aplicados directamente a un conjunto de datos o llamados desde su propio código Java. Weka contiene herramientas de pre-procesamiento de datos, clasificación, regresión, clustering, reglas de asociación y visualización. También es muy apropiada para el desarrollo de nuevos sistemas de aprendizaje automático.

El paquete Weka contiene una colección de herramientas de visualización y algoritmos para análisis de datos y modelado predictivo, unidos a una interfaz gráfica de usuario para acceder fácilmente a sus funcionalidades. los puntos fuertes de Weka son:

- Está disponible libremente bajo la licencia pública general de GNU.
- Es muy portable porque está completamente implementado en Java y puede correr en casi cualquier plataforma.
- Contiene una extensa colección de técnicas para preprocesamiento de datos y modelado.
- Es fácil de utilizar por un principiante gracias a su interfaz gráfica de usuario.

Weka soporta varias tareas estándar de minería de datos, especialmente, preprocesamiento de datos, clustering, clasificación, regresión, visualización, y selección. Todas las técnicas de Weka se fundamentan en la asunción de que los datos están disponibles en un fichero plano o una relación, en la que cada registro de datos está descrito por un número fijo de atributos (normalmente numéricos o nominales, aunque también se soportan otros tipos). Weka también proporciona

acceso a bases de datos vía SQL gracias a la conexión JDBC (*Java Database Connectivity*) y puede procesar el resultado devuelto por una consulta hecha a la base de datos. No puede realizar minería de datos multi-relacional, pero existen aplicaciones que pueden convertir una colección de tablas relacionadas de una base de datos en una única tabla que ya puede ser procesada con Weka .

La mayoría de herramientas de minería de datos están basadas en esta plataforma por su fácil de implementar debido a la cantidad y claridad de su documentación.

Sistema Operativo: Win y Linux

Algoritmos Implementados:

- ✓ Cobweb
- ✓ DBScan
- ✓ EM
- ✓ Farthest Firts
- ✓ Filterer Clusterer
- ✓ MakeDensityBaseClusterer
- ✓ OPTICS
- ✓ Simple KMeans
- ✓ X-Means

4.1.13 Lenguaje R.

R es un conjunto de programas integrados para manejo de datos, simulaciones, cálculos y realización de gráficos. Es además un lenguaje de programación orientado a objetos. R es una implementación libre, independiente, open-source. Contiene diversos algoritmos de clustering y variadas funciones de medir distancias, además de tener diferentes mecanismos para evaluar los resultados de los algoritmos.

El proyecto R fue iniciado por Robert Gentleman y Ross Ihaka (de donde se deriva “R”) del Statistics Department, University of Auckland, en 1995. Actualmente R es mantenido por un grupo internacional de desarrolladores *voluntarios*: Core development team.

Algoritmos Implementados:
KMeans
Clustering Jerarquico
EM
Rock
DBSCAN
Otros

4.2 HERRAMIENTAS COMERCIALES

4.2.1 DB2 Intelligent Miner for Data

DB2 Intelligent Miner for Data [97] *es la plataforma de inteligencia empresarial para el análisis en toda la organización.* Desarrollado por IBM. Proporciona un conjunto de herramientas que forman un marco de trabajo para la extracción de datos. Admite el proceso iterativo y permite el procesamiento de datos, el análisis estadístico y la visualización de resultados como suplemento de su amplia variedad de métodos de extracción. Utiliza algoritmos de extracción garantizados, ya sea individualmente o combinados, para resolver numerosos problemas de negocios y obtener resultados comerciales mensurables. Proporciona una solución ampliable, centrada en las áreas clave de la extracción a gran escala, como por ejemplo, los grandes volúmenes de datos, la extracción paralela de datos, las operaciones de extracción a largo plazo y la mejora de los algoritmos de extracción. Incluye un interfaz de programación de aplicaciones que permite el desarrollo de aplicaciones de extracción personalizadas adaptadas a cada sector específico [97].

Sistema Operativo: Win y Linux

Algoritmos de Utilizados:

- ✓ Clustering Demográfico
- ✓ Clustering Particional

4.2.2 Oracle Data Mining

Implementa una variedad de algoritmos de minería de datos dentro de la base de datos relacional Oracle. Estas implementaciones se han integrado en el núcleo de base de datos Oracle, y opera en forma nativa los datos almacenados en las tablas de base de datos relacional [Ora]. Esto elimina la necesidad de extracción o transferencia de datos independiente del análisis de minería. La plataforma de base de datos relacional es apalancamiento para gestionar modelos de forma segura y eficiente ejecutar consultas SQL en grandes volúmenes de datos. El sistema está organizado en torno a unas operaciones genéricas que proporcionan una interfaz unificada de funciones de minería de datos. Estas operaciones incluyen funciones para crear, aplicar, probar y manipular los modelos de minería de datos. Los modelos son creados y almacenados como objetos de bases de datos, y su gestión se realiza dentro de la base de datos, similar a las tablas, vistas, índices y otros objetos de base de datos.

Sistema Operativo: Todos

Algoritmos de Utilizados:

- ✓ Enhanced k-means (EKM).
- ✓ Orthogonal Partitioning Clustering (O-Cluster).

4.2.3 SQL Server Analysis Services – Minería de datos

Microsoft SQL Server Analysis Services [92] contiene las características y herramientas necesarias para crear complejas soluciones de minería de datos.

- Un conjunto de algoritmos de minería de datos estándar del sector.
- El Diseñador de minería de datos, que sirve para crear, administrar y examinar modelos de minería de datos para, a continuación, crear predicciones a partir de dichos modelos.
- El lenguaje DMX (Extensiones de minería de datos), que sirve para administrar modelos de minería de datos y crear complejas consultas predictivas.

Analysis Services proporciona una amplia gama de herramientas que puede usar para generar soluciones de minería de datos, tanto con datos de cubo como con datos relacionales. Además, SQL Server proporciona las siguientes herramientas de Business Intelligence: Integration Services, que le ayuda a generar procesos ETL para la limpieza de datos y el procesamiento o actualización de modelos, y Reporting Services, que le ayuda a presentar predicciones y permitir a los usuarios explorar los datos.

Una vez completado el modelo, puede implementarlo en otro servidor para que los usuarios puedan realizar análisis y predicciones ad hoc mediante los modelos almacenados. Puede tener acceso a los modelos de minería de datos a través de clientes personalizados, incluyendo servicios web, o usando las aplicaciones de Microsoft Office, como los complementos de minería de datos para Excel. En SQL Server 2008, la minería de datos es eficaz y accesible, y está integrada con las herramientas preferidas de los usuarios para el análisis y la creación de informes.

Sistema Operativo: Win

Algoritmos de Utilizados:

- ✓ EM
- ✓ K-means

4.2.4 SPSS

SPSS [98] se deriva de Statistical Package for the Social Sciences, o también de la nombre de la empresa que lo distribuye Statistical Product and Service Solutions. Es un software muy completo que está provisto de diferentes, técnicas estadísticas, además de soportar varios tipos de fuentes de datos y está provisto de diferentes formas de ver los datos como estadísticas descriptivas, pruebas t, regresión lineal y pruebas no paramétricas. Además permite al usuario ingresar una serie de sentencias para realizar pruebas de nuevos algoritmos y acondicionamiento de los datos.

Sistema Operativo: Win, Linux y MacOS .

Algoritmos de clustering implementados:

- ✓ Clustering Jerarquico
- ✓ TwoStep Cluster
- ✓ Quick Cluster.

5. METODOLOGIA, LENGUAJES Y HERRAMIENTAS PARA EL DESARROLLO DE RASEMUS

Entre las principales herramientas de desarrollo podemos encontrar al lenguaje de programación Java que incluye la biblioteca grafica Swing, el entorno de desarrollo NetBeans que es uno de los mejores editores para java y de licencia libre, además se menciona la herramienta Tariy, ya que tener licencia GNU que permitió reutilizar los módulos de interfaz grafica y de conexión a bases de datos, haciendo alguna modificaciones para el uso en Rasemus.

A continuación se explican brevemente los conceptos preliminares de estas herramientas para ser tenidas en cuenta en la posterior descripción de la implementación.

5.1 LENGUAJE DE PROGRAMACIÓN JAVA.

Java es un lenguaje de programación orientado a objetos desarrollado por James Gosling y sus compañeros de Sun Microsystems al inicio de la década de 1990. A diferencia de los lenguajes de programación convencionales, que generalmente están diseñados para ser compilados a código nativo, Java es compilado en un bytecode que es ejecutado (usando normalmente un compilador JIT), por una maquina virtual Java. El lenguaje Java se crea con cinco objetivos principales:

1. Deberá usar la metodología de la programación orientada a objetos.

2. Deberá permitir la ejecución de un mismo programa en múltiples sistemas operativos.

3. Deberá incluir por defecto soporte para trabajo en red.

4. Deberá diseñarse para ejecutar código en sistemas remotos de forma segura.

5. Deberá ser fácil de usar y tomar lo mejor de otros lenguajes orientados a objetos, como C++.

5.1.1 Características Principales:

Orientado a Objetos

La primera característica, orientado a objetos (”OO”), se refiere a un método de programación y al diseño del lenguaje. Aunque hay muchas interpretaciones para OO, una primera idea es diseñar el software de forma que los distintos tipos de datos que use están unidos a sus operaciones. Así los datos y el código (funciones o métodos) se combinan en entidades llamadas objetos. Un objeto puede verse como un paquete que contiene el “comportamiento” (el código) y el “estado” (datos).

El principio es separar aquello que cambia de las cosas que permanecen inalterables. Frecuentemente, cambiar una estructura de datos implica un cambio en el código que opera sobre los mismos, o viceversa. Esta separación en objetos coherentes e independientes ofrece una base más estable para el diseño de un sistema software. El objetivo es hacer que grandes proyectos sean fáciles de gestionar y manejar, mejorando como consecuencia su calidad y reduciendo el número de proyectos fallidos. Otra de las grandes promesas de la programación orientada a objetos es la creación de entidades más Genéricas (objetos) que permitan la reutilización del software entre proyectos, una de las premisas fundamentales de la Ingeniería del Software. Un objeto genérico ”cliente”, por ejemplo, Deberá en teoría tener el mismo conjunto de comportamiento en diferentes proyectos, sobre todo cuando estos coinciden en cierta medida, algo que suele suceder en las grandes organizaciones. En este sentido, los objetos Podría verse como piezas reutilizables que pueden emplearse en múltiples proyectos distintos, posibilitando así a la industria del software a construir proyectos de envergadura empleando componentes ya existentes y de comprobada calidad; conduciendo esto finalmente a una reducción drástica del tiempo de desarrollo. Podemos usar como ejemplo de objeto el aluminio. Una vez definidos datos (peso, maleabilidad, etc.), y su “comportamiento” (soldar dos piezas, etc.), el objeto “aluminio” puede ser reutilizado en el campo de la construcción, del automóvil, de la aviación, etc.

Independencia de la plataforma

La segunda característica, la independencia de la plataforma, significa que programas escritos en el lenguaje Java pueden ejecutarse igualmente en cualquier tipo de hardware. Es lo que significa ser capaz de escribir un programa una vez y que pueda ejecutarse en cualquier dispositivo, tal como reza el axioma de Java, ”write once, run everywhere”. Para ello, se compila el código fuente escrito en lenguaje Java, para generar un código conocido como “bytecode” (específicamente Java bytecode) que son instrucciones maquina simplificadas especificas de la plataforma Java. Esta pieza esta “a medio camino” entre el código fuente y el código maquina que entiende el dispositivo destino. El bytecode

es ejecutado entonces en la maquina virtual (VM), un programa escrito en código nativo de la plataforma destino (que es el que entiende su hardware), que interpreta y ejecuta el código. Además, se suministran bibliotecas adicionales para acceder a las características de cada dispositivo (como los gráficos, ejecución mediante hebras o threads, la interfaz de red) de forma unificada. Se debe tener presente que, aunque hay una etapa explicita de compilación, el bytecode generado es interpretado o convertido a instrucciones maquina del código nativo por el compilador JIT (Just In Time).

El recolector de basura

Un argumento en contra de lenguajes como C++ es que los programadores se encuentran con la carga añadida de tener que administrar la memoria de forma manual. En C++, el desarrollador debe asignar memoria en una zona conocida como heap (monticulo) para crear cualquier objeto, y posteriormente desalojar el espacio asignado cuando desea borrarlo. Un olvido a la hora de desalojar memoria previamente solicitada, o si no lo hace en el instante oportuno, puede llevar a una fuga de memoria, ya que el sistema operativo piensa que esa zona de memoria está siendo usada por una aplicación cuando en realidad no es así un programa mal diseñado Podría consumir una cantidad desproporcionada de memoria. Además, si una misma Región de memoria es desalojada dos veces el programa puede volverse inestable y llevar a un eventual cuelgue.

En Java, este problema potencial es evitado en gran medida por el recolector automático de basura (o automatic garbage collector). El programador determina cuando se crean los objetos y el entorno en tiempo de ejecución de Java (Java runtime) es el responsable de gestionar el ciclo de vida de los objetos. El programa, u otros objetos pueden tener localizado un objeto mediante una referencia a este (que, desde un punto de vista de bajo nivel es una dirección de memoria). Cuando no quedan referencias a un objeto, el recolector de basura de Java borra el objeto, liberando así la memoria que ocupaba previniendo posibles fugas (ejemplo: un objeto creado y únicamente usado dentro de un método solo tiene entidad dentro de este; al salir del método el objeto es eliminado), aun así es posible que se produzcan fugas de memoria si el código almacena referencias a objetos que ya no son necesarios es decir, pueden aun ocurrir, pero en un nivel conceptual superior. En definitiva, el recolector de basura de Java permite una fácil creación y eliminación de objetos, mayor seguridad y frecuentemente más rápida que en C++.

La recolección de basura de Java es un proceso prácticamente invisible al desarrollador. Es decir, el programador no tiene conciencia de cuando la recolección de basura tendrá lugar, ya que esta no tiene necesariamente que guardar relación con las acciones que realiza el código fuente. Debe tenerse en cuenta que la memoria es solo uno de los muchos recursos que deben ser gestionados.

5.2 ENTORNO DE DESARROLLO NETBEANS.

NetBeans se refiere a una plataforma para el desarrollo de aplicaciones de escritorio usando Java y a un Entorno integrado de desarrollo (IDE) desarrollado usando la Plataforma NetBeans. La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un modulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como modulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.

La Plataforma NetBeans es un framework reusable que simplifica el desarrollo de otras aplicaciones de escritorio. Cuando se ejecuta una aplicación basada en la Plataforma NetBeans, la plataforma ejecuta la clase Main. Los módulos disponibles están localizados y puestos en un registro en memoria, y son ejecutadas las tareas de inicialización de los módulos. Generalmente el código de un modulo es cargado en memoria solo cuando se necesita. La plataforma ofrece servicios comunes a las aplicaciones de escritorio, permitiéndole al desarrollador enfocarse en la lógica específica de su aplicación. Entre las características de la plataforma están:

1. Administración de las interfaces de usuario (ej. menús y barras de herramientas).

2. Administración de las configuraciones del usuario.

3. Administración del almacenamiento (guardando y cargando cualquier tipo de dato).

4. Administración de ventanas.

5. Framework basado en asistentes (diálogos pasos a paso).

5.3 CONTROLADOR JDBC.

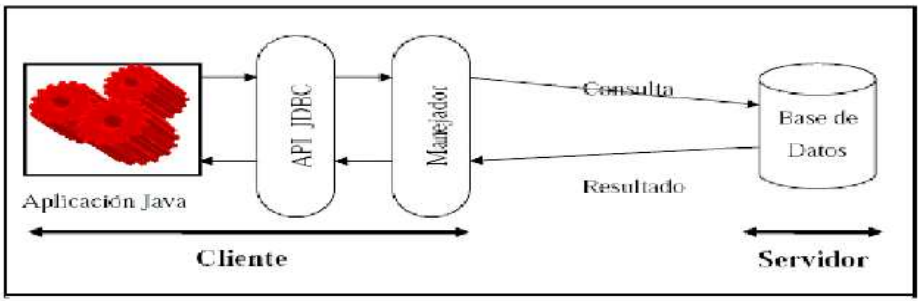
JDBC es el acrónimo de Java Database Connectivity, un API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java independientemente del sistema de operación donde se ejecute o de la base de datos a la cual se accede utilizando el dialecto SQL del modelo de base de datos que se utilice.

El API JDBC se presenta como una colección de interfaces Java y métodos de gestión de manejadores de conexión hacia cada modelo especifico de base de datos. Un manejador de conexiones hacia un modelo de base de datos en particular es un conjunto de clases que implementan las interfaces Java y que utilizan los métodos de registro para declarar los tipos de localizadores a base de datos (URL) que pueden manejar. Para utilizar una base de datos particular, el usuario ejecuta su programa junto con la biblioteca de conexión apropiada al modelo de su base de datos, y accede a ella estableciendo una conexión, para ello provee en localizador a la base de datos y los parámetros de conexión específicos. A partir de allí puede realizar cualquier tipo de tareas con la base de datos a las que tenga permiso: consultas, actualizaciones, creado modificado y borrado de tablas, ejecución de procedimientos almacenados en la base de datos, etc. Cada base de datos emplea un protocolo diferente de comunicación, protocolos que normalmente son propietarios. El uso de un manejador, una capa intermedia entre el código del desarrollador y la base de datos, permite independizar el código Java que accede a la BD del sistema de BD concreto a la que estamos accediendo, ya que en nuestro código Java emplearemos comandos estándar, y estos comandos serian traducidos por el manejador a comandos propietarios de cada sistema de BD concreto. Si queremos cambiar el sistema de BD que empleamos lo único que deberemos hacer es reemplazar el antiguo manejador por el nuevo, y seremos capaces de conectarnos la nueva BD.

Manejador de Protocolo Nativo (tipo 4).

El manejador traduce directamente las llamadas al API JDBC al protocolo nativo de la base de datos. Es el manejador que tiene mejor rendimiento, pero está más ligado a la base de datos que empleemos que el manejador tipo JDBCNet, donde el uso del servidor intermedio nos da una gran flexibilidad a la hora de cambiar de base de datos. Este tipo de manejadores también emplea Tecnología 100 % Java.

Figura 52: Controlador JDBC tipo 4



5.4 BIBLIOTECA GRAFICA SWING DE JAVA.

Swing es una biblioteca grafica para Java que forma parte de las Java Foundation Classes (JFC). Incluye componentes para interfaz grafica de usuario tales como cajas de texto, botones, listas desplegables y tablas. Las Internet Foundation Classes (IFC) eran una biblioteca grafica para el lenguaje de programación Java desarrollada originalmente por Netscape y que se publico en 1996. En 1997, Sun Microsystems y Netscape Communications Corporation anunciaron su intención de combinar IFC con otras Tecnología de las Java Foundation Classes. Además de los componentes ligeros suministrados originalmente por la IFC, Swing introdujo un mecanismo que permito que el aspecto de cada componente de una aplicación pudiese cambiar sin introducir cambios sustanciales en el código de la aplicación. La introducción de soporte ensamblable para el aspecto permitió a Swing emular la apariencia de los componentes nativos manteniendo las ventajas de la independencia de la plataforma.

Algunos de los componentes utilizados en el desarrollo de este proyecto y pertenecientes a la biblioteca Swing se explican a continuación:

- ✓ **JComponent.** La clase base para todos los componentes de Swing exceptuando los contenedores de nivel superior. Para utilizar un componente que herede de JComponent, se debe poner el componente en una jerarquía de la contención cuya raíz sea un contenedor de nivel superior Swing tal como Ventanas o Paneles. Los contenedores de nivel superior son los componentes especializados que proporcionan un lugar para que otros componentes de Swing puedan pintarse.

- ✓ **JFrame.** Un JFrame es una ventana de nivel superior con un titulo y un borde. El tamaño del JFrame incluye cualquier área señalada por los bordes donde puede ser incluido uno más componentes de Swing.

- ✓ **JPanel**. El JPanel es la clase más simple del contenedor. Un JPanel proporciona el espacio en el cual una aplicación puede unir cualquier otro componente, incluyendo otros paneles.

- ✓ **JTabbedPane.** Un componente que deja a usuario cambiar entre un grupo de componentes pulsando en una etiqueta con un titulo dado y/o un icono. Las etiquetas y los componentes son agregados a un objeto de TabbedPane usando los métodos del addTab e insertTab. Una etiqueta es representada por un índice que corresponde a la posición que fue agregado adentro, donde la primera etiqueta tiene un índice igual a 0 y la etiqueta pasada tiene un índice igual a la cuenta de la etiqueta menos 1.

- ✓ **JSplitPane**. JSplitPane se utiliza para dividir dos (y solamente dos) componentes. Los dos componentes se pueden entonces volver a clasificar según el tamaño recíprocamente por el usuario. La información sobre cómo usar JSplitPane está en cómo utilizar los paneles partidos en la clase particular de Java. Se pueden incluir nuevos componentes en cada una de las divisiones de un JSplitPanne. Los dos componentes en un panel partido se pueden alinear a la izquierda y a la derecha usando JSplitPane.HORIZONTAL SPLIT, o en la parte superior o inferior de la ventana con JSplitPane.VERTICAL SPLIT.

- ✓ **JScrollPane**. JScrollPane proporciona una vista deslizante de un componente ligero. Un JScrollPane maneja un viewport, barras de desplazamientos verticales y horizontales opcionales, y viewports opcionales del título de la fila y de columna. El viewport, o punto de vista, proporciona una ventana sobre una fuente de datos, por ejemplo, un archivo de texto o una imagen sobre la cual se va a deslizar.

- ✓ **JFileChooser**. Los seleccionadores de archivos proporcionan una interfaz grafica de usuario para navegar el sistema de ficheros, y después elegir un archivo o un directorio de una lista o incorporar el nombre de un archivo o un directorio. Para exhibir un seleccionador de archivo, se utiliza generalmente el JFileChooser para mostrar un dialogo modal que contiene el seleccionador de archivo. Otra manera de presentar un seleccionador es agregar una instancia.

- ✓ **JTable**. Con la clase de JTable se puede exhibir una tabla de datos, permitiendo opcionalmente que el usuario edite el contenido de las celdas que contienen los datos. JTable no contiene ni deposita datos; es simplemente una vista de los datos.

- ✓ **JTree**. Con la clase JTree, puedes exhibir datos jerárquicos. Un objeto JTree no contiene realmente los datos sino que proporciona simplemente una vista de ellos. Como cualquier componente no trivial del Swing, el árbol consigue los datos conectándose a un modelo de los datos.

- ✓ **JTextArea**. Un JTextArea es un área multilinea que exhibe el texto plano y que permite opcionalmente que el usuario corrija el texto. Este es un componente ligero que provee de compatibilidad a la hora de introducción o despejar pequeñas cantidades de información.

- ✓ **JSpinner**. Esta clase provee en una sola línea la posibilidad al usuario de entrar o seleccionar un valor de una secuencia pedida. Los JSpinner proporciona típicamente un par de los botones minúsculos de flechas para cambiar entre los elementos de la secuencia. Las teclas de flecha arriba/abajo del teclado también completan un ciclo a través de los elementos. El usuario puede también mecanografiar el valor directamente en un JSpinner.

- ✓ **JComboBox**. Un componente que combina un botón o un campo editable y una lista desplegable. El usuario puede seleccionar un valor de la lista, que aparece a petición del usuario. Si la caja de texto es editable, entonces se incluye un campo en el cual el usuario pueda mecanografiar un valor.

- ✓ **JRadioButton**. Una puesta en práctica de un botón de radio. Este es un componente que puede ser seleccionado o deseleccionado, y que exhibe su estado al usuario. Utilizado con un objeto ButtonGroup crea un grupo de botones en los cuales solamente un botón puede ser seleccionado a la vez.

5.5 JFREECHART

JFreeChart es una biblioteca gráfica de Java 100% libre, la cual facilita a los desarrolladores mostrar gráficos de calidad profesional en sus aplicaciones.
El amplio conjunto de características de JFreeChart incluye:

- ✓ Una consistente y bien documentada API, dando soporte para una amplia gama de tipos de gráficos.

- ✓ Un diseño flexible que es fácil de extender, abarcando aplicaciones tanto del lado del servidor como del lado del cliente.

- ✓ Soporte para muchos tipos de salida, incluyendo los componentes Swing, archivos de imagen (incluyendo PNG y JPEG), gráficos vectoriales y formatos de archivo (incluyendo PDF, EPS y SVG).

- ✓ JFreeChart es “código abierto” o, más concretamente, software libre. Se distribuye bajo los términos de la Licencia LGPL (GNU Lesser General Public Licence), que permite su uso en aplicaciones propietarias.

El proyecto JFreeChart fue fundado en febrero del año 2000 por David Gilbert. Hoy en día, JFreeChart es utilizada por aproximadamente 40.000 a 50.000 desarrolladores. El proyecto sigue siendo administrado por Gilbert, con contribuciones de una amplia comunidad de desarrolladores.

5.6 ITEXT

iText [90][105] es una biblioteca Open Source para crear y manipular archivos PDF, RTF, y HTML en Java. Fue escrita por Bruno Lowagie, Paulo Soares, y otros; está distribuida bajo la Mozilla Public License con la LGPL como licencia alternativa.

El mismo documento puede ser exportado en múltiples formatos, o múltiples instancias del mismo formato. Los datos pueden ser escritos a un fichero o, por ejemplo, desde un servlet a un navegador web.

Más recientemente, ha sido extendida a una biblioteca PDF de propósito general, capaz de rellenar formularios, mover páginas de un PDF a otro, y otras cosas. Estas extensiones son a menudo mutuamente excluyentes. Una clase te permite rellenar en formularios, mientras una clase diferente e incompatible hace posible copiar páginas de un PDF a otro.

El soporte de PDF de iText es, sin embargo, bastante extensivo. Esto soporta firmas basadas en PKI de PDF, cifrado de 40-bit y 128-bit, corrección de colores, PDF/X, gestión de colores por perfiles ICC, y es anfitriona de otras características.

5.7 JEXCELAPI

Java Excel Api (JExcelApi) [JExcelApi] es un proyecto maduro [83]. Es una API desarrollada bajo código abierto que permite a los desarrolladores de java leer, escribir y modificar hojas de cálculo de archivos con formato .xls de una forma dinámica y simple.

Algunas Características

- ✓ Lee archivos de Excel 95, 97, 2000, XP y 2003
- ✓ Lee y escribe fórmulas en formato Excel 97, 2000, XP y 2003
- ✓ Genera hojas de cálculo en formato Excel 2000
- ✓ Soporta tipos de letra, número y la fecha
- ✓ Soporta sombreado, limites y de coloración de las celdas
- ✓ Manejo hojas de cálculo.
- ✓ Apoya la inserción y la copia de imágenes en hojas de cálculo

5.8 TARIYKDD

TariyKDD, una es herramienta genérica para el Descubrimiento de Conocimiento en Bases de Datos débilmente acoplada con el SGBD PostgreSQL. TariyKDD comprende cuatro módulos que cubren la conexión, a archivos planos y bases de datos relacionales, un modulo de utilidades con clases y bibliotecas comunes, un modulo kernel que reúne las etapas de preprocesamiento, minería y visualización, y el modulo de interfaz grafica de usuario.

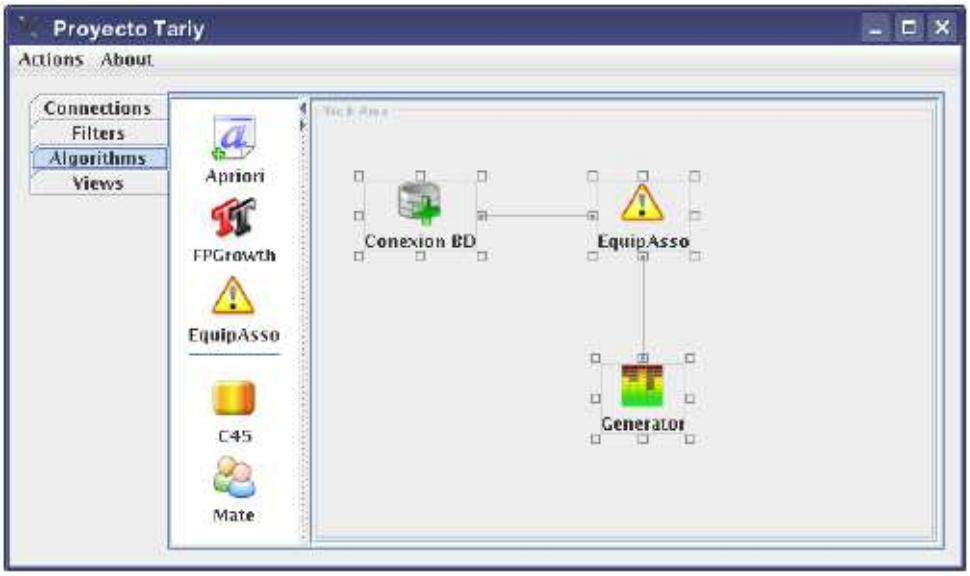
TariyKdd fue desarrollada bajo licencia pública GNU que permite la reutilización de sus módulos para la implementación en otras herramientas. A continuación se presentan los módulos *KnowledgeFlow* y *DBConnection*. El primero contiene las formas utilizadas en la Interfaz principal de la aplicación. En el paquete *DBConnection* contiene todo lo concerniente a realizar una conexión a una base de datos, desde permitir la conexión hasta maneja los datos en memoria.

5.8.1 Paquete KnowledgeFlow

Se hablara primero de la implementación del paquete *KnowledgeFlow*. La interfaz principal de la aplicación está contenida dentro de la clase *Chooser*, cuya implementación se muestra en la figura 53 y a su vez implementa el uso de diversas clases propias de la biblioteca grafica *Swing* de Java como *JTabbedPanne*, *JsplitPane*, *JscrollPane*, *JLabel* así como extensiones de la clase *JPanel* donde se implementan funcionalidades propias a la aplicación como un área de trabajo donde tendrá lugar la construcción de un experimento KDD y donde, a través de la metodología de Arrastrar y Soltar (*Drag'n Drop*), se dispondrán los iconos que representan una determinada tarea dentro del proceso. Se extiende también *JComponent* donde se implementan funcionalidades de los iconos que representan cada acción dentro de la aplicación.

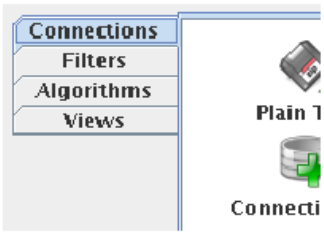
Dentro de la interfaz principal se pueden distinguir algunas secciones como son un Selector, en el cual se permite escoger una etapa dentro del proceso KDD, un panel, donde se despliegan los iconos asociados a una determinada etapa y una Área de Trabajo, donde se organizan los iconos seleccionados y se construye un experimento de descubrimiento de conocimiento.

Figura 53: Clase *Chooser*. Interfaz principal de la aplicación TariyKDD



Cada una de estas partes se constituye como una nueva clase en el proyecto, que se gestionan con instancias propias de Java las cuales se mencionaron anteriormente. Con un *JTabbedPanne* se monta la selección de un determinado proceso desplegando etiquetas que identifican a cada uno de ellos, ”*Connections*”, para escoger una conexión hacia un conjunto de datos especifica, puede ser a una base de datos o cargando un archivo plano, ”*filters*”, donde se da la opción de escoger un determinado filtro que modificara el conjunto de datos que se haya cargado, ”*Algorithms*”, sección en la cual se escoge el algoritmo oportuno para realizar las tareas de minería de datos como tal y ”*Views*”, donde se despliegan opciones de visualización para organizar y mostrar al usuario los resultados obtenidos. Un detalle de la implementación de esta estructura se muestra en la figura 54.

Figura 54: Etiquetas dentro de *Chooser* para la selección de etapas



Al interactuar con las etiquetas del *JTabbedPanne* se verá modificado el contenido de la clase *Container*, la cual es una extensión de *JSplitPanne* y divide este

componente en dos, izquierda y derecha. En el izquierdo se cargara un panel correspondiente a cada etapa seleccionada con el *JTabbedPanne*, los posibles paneles que se cargan se visualizan en la figura 55, y en el derecho se carga una instancia de la clase *MyCanvas*, que provee la funcionalidades de *Drag'n Drop* (Arrastrar y Soltar) entre los iconos involucrados en un experimento.

Figura 55: Paneles para cada etapa del proceso KDD

Cada panel extiende a *JPanel* y está conformado por instancias de la clase *JLabel*, cada una de las cuales representara un icono perteneciente a esa etapa. En la instanciación de cada *JLabel* se carga una imagen alusiva a cada icono y un texto que lo identifica dentro del panel. Las imágenes correspondientes a cada icono, y en general todas las imágenes utilizadas en la herramienta, se cargan directamente del *Classpath* de la aplicación dentro del paquete *images*. Es importante aclarar que en este momento también se asigna el nombre de cada *JLabel*, en su propiedad *name* a través del método *setName*, ese mismo nombre será relacionado posteriormente para identificar el icono seleccionado por el usuario desde el panel, luego, cada *JLabel* es adicionado a su respectivo panel.

Un fragmento de código donde es asignado los valores para cada icono se muestra en la figura 56

Figura 56: asignación de valores a Iconos

```
jLabel.setIcon(new ImageIcon(getClass().getResource(
                                        "/images/connection.png")));
jLabel.setText(" Connection DB ");
jLabel.setName("connection");
jLabel.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
jLabel.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
panelConnections.add(jLabel);
```

Se puede notar en la primera línea como se asigna una imagen al *JLabel* utilizando el método *getClass().getResource*(Ruta de la imagen). La clase *MyCanvas* es la encargada de implementar las funcionalidades de *Drag'n Drop* para los iconos involucrados en un determinado experimento. Al seleccionar con el mouse un determinado icono desde un panel, el *JLabel* asociado a ese icono es capturado escaneando los eventos del mouse cuando un icono es presionado y posteriormente liberado. El siguiente fragmento de Figura 57 ilustra cómo es capturado un *JLabel* al ser presionado con el mouse.

Figura 57: Captura de *JLabel*

```
JLabel pressed;
container.addMouseListener(new java.awt.event.MouseAdapter() {
  public void mousePressed(java.awt.event.MouseEvent evt) {
    pressed = container.findComponentAt(evt.getPoint());
  }
});
```

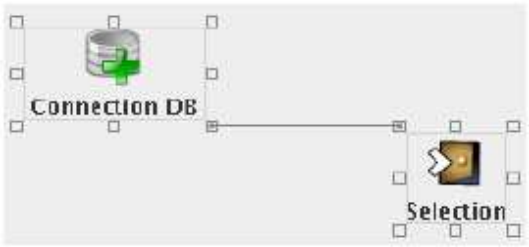
Al adicionar un *MouseListener* al objeto *container*, el *JSplitPanne* que contiene los paneles y el área de trabajo, este captura el evento *pressed* del mouse cuando este ha sido presionado. En ese momento, el evento es capturado y a través del método *getPoint* tenemos la posición dentro del contenedor donde fue generado el click. El método *findComponent*() devolverá el componente encontrado en el punto donde fue presionado el mouse.

Cuando el mouse se libera, este evento es igualmente capturado y se procede a identificar cual fue el icono seleccionado desde el panel para desplegarlo correctamente sobre el objeto *MyCanvas*. Esto se efectúa al consultar la propiedad *name* del objeto *pressed* que fue capturado durante el evento anterior.

Dependiendo del nombre del componente se procede a instanciar un objeto de la clase *Icon*.

La clase *Icon* fue construida extendiendo a un *JPanel* y contiene una instancia de la clase *JLabel*, que contiene la imagen y el texto asociados a ese icono, y un total de 8 instancias de la clase Conector, esta clase fue diseñada para identificar secciones dentro del icono donde el usuario pueda hacer un click y relacionar un icono con otro. En la figura 58 se muestra la inclusión de 2 instancias de la clase *Icon* y la relación establecida entre ellos a través de sus conectores.

Figura 58: Implementación de la clase *Icon*



Se implementa dentro de esta clase la funcionalidad un menú desplegable donde se pueda incluir funciones propias para cada icono y la inclusión de una animación que se activa cuando el icono está realizando un proceso determinado. Estas funcionalidades serán heredadas a todas las clases que extiendan de la clase *Icon*.

El menú desplegable se logra al usar las clases *JPopupmenu* y *JmenuItem*. Por defecto, todas las instancias de *Icon* y las clases que hereden de el tendrán implementada la opción *Delete*, que borrara el icono del área de trabajo y descargara la clase *Icon* correspondiente de la clase *MyCanvas* que la contiene. Para ello, se instancia un objeto de la clase *JmenuItem* y se setea su propiedades de name a ”*Delete*”, posteriormente se procede a adicionar esta instancia de *JmenuItem* al *JPopupmenu* asociado a la clase *Icon*. Cuando un usuario seleccione la opción *Delete* se dispara el método *mnuDeleteActionPerformed* que se encarga de descargar las conexiones que tenga asociado este icono con otros

iconos y borrar el componente del área de trabajo. Nuevas adiciones al menú desplegable pueden ser hechas al instanciar objetos *JmenuItem* e incluirlos dentro del *JPopupmenu* de la clase *Icon*.

La clase Conector fue construida extendiendo *JComponent* ya que debía proveer facilidades de captura de los eventos del mouse y ser dibujado de manera especial cuando esté disponible y diferente cuando ya haya sido seleccionado. A continuación se muestra un fragmento de Figura 59 donde se ilustra el dibujado de esta clase al sobrecargar el método *paint* de la clase *Jcomponent*.

Figura 59: Dibujado de la clase conector

```
public synchronized void paint(Graphics g){
    g.setColor(Color.Blue);
    g.drawRect(0, 0, 6, 6);
    if(selected){
        g.fillRect(2, 2, 3, 3);
    }
}
```

Vemos que al método le llega una instancia g del la clase *Graphics* que será el objeto donde podremos dibujar. A través de los métodos de esta clase podemos escoger colores para el conector y dibujar un rectángulo de 7x7 pixels que representara el área del conector que puede ser pulsada por el usuario para su selección y dibujaría un rectángulo relleno en el centro del conector si este ha sido ya seleccionado.

La clase *Icon* como tal no es incluida directamente sobre la clase *MyCanvas*, o área de trabajo. Existen una serie de clases que extiende a la clase *Icon* y se corresponden con cada tarea desempeñada dentro del proceso KDD. Existen un total de 7 extensiones a la clase *Icon* y estas son *DBConnectioIcon, fileIcon, filterIcon, AssociationIcon, ClassificationIcon, RulesIcon y TreeIcon* asociadas a la conexión con una base de datos, conexiones con archivos planos, filtros para la selección y preprocesamiento de un conjunto de datos, algoritmos de asociación, algoritmos de clasificación, visualización de resultados de reglas de asociación y visualización de arboles de decisión respectivamente. Estas 7 clases, junto con otras que apoyan la tarea que cumplen, están contenidas en 7 paquetes dentro del paquete *Icons* que será explicado posteriormente y que hace parte a su vez del paquete *gui*.

Para cada una de las clases heredadas se maneja de manera independiente el contenido de su *JPopupmenu* así como la imagen y el texto asociados a cada tipo de icono.

La clase *MyCanvas* será la encargada de gestionar las conexiones entre los iconos y su movimiento sobre el área de trabajo. Esta clase extiende un *JPanel* y monitorea los eventos del mouse referentes a los *clicks* del usuario. Los eventos que tiene contemplados son *MousePressed* (mouse presionado), *MouseDragged* (click sostenido), *MouseReleased* (mouse liberado), *MouseClicked* (un click sencillo).

Para el evento *MousePressed*, se identifica si el click fue sobre un icono o sobre uno de sus conectores. Si fue sobre un icono este es almacenado en la variable *selectedIcon*, pero si fue sobre un conector se almacena en la variable *selectedConector*. Para esta tarea se utiliza el método *findComponentAt(MouseEvent*) que se explico anteriormente. Al final de este método, y en general todos los métodos que involucran cambios sobre el área de trabajo, se llama al método *repaint*()que se encarga de redibujar la clase *MyCanvas* invocando directamente el método *paint(Graphics g)* de esta clase y que se explicara posteriormente. Se presenta el siguiente fragmento de código figura 60 para explicar el método *MousePressed*.

Figura 60: Click sobre un icono

```
private void formMousePressed(java.awt.event.MouseEvent evt) {
    Component press = this.findComponentAt(evt.getPoint());
    //Si se presiono un Icono
    if(press instanceof Icon){
        selectedIcon = (Icon)press;
    }
    //Si se presiono un Conector
    else if(press instanceof Conector){
        selectedConector = (Conector)press;
    //Se redibuja el area de trabajo para actualizar los cambios
    repaint();
}
```

Para el evento *MouseDragged* se identifica cual fue el componente seleccionado en el evento *MousePressed*, si se trata de un icono este método se encarga de redibujarlo a medida que se mueve el mouse pero si se trata de un conector, se traza una línea entre el conector seleccionado y el puntero del mouse a medida que este se mueve. Estas dos acciones se realizan en el método *paint(Graphics g)* de la clase *MyCanvas*. Se muestra el siguiente fragmento de figura 61 para ilustrar lo hecho en este método.

Figura 61: Evento Arrastrar y soltar

```
private void formMouseDragged(java.awt.event.MouseEvent evt) {
  //Si un Icono fue ya selecconado
  if(selectedIcon != null){
    //Se captura la posicion del puntero
    int x = evt.getX();
    int y = evt.getY();
    //Se calcula la nueva posicion del icono
    //a partir de la posicion del mouse
    selectedIcon.setLocation(x - selectedIcon.getWidth() / 2,
                             y - selectedIcon.getHeight() /2);
    //Si un Conector fue seleccionado
  } else if(selectedConector != null){
    //se capturan las coordenadas del puntero del mouse
    xMouse = evt.getX();
    yMouse = evt.getY();
  }
  //Se redibuja la forma para actualizar los cambios
  //Redibujar el Icono o trazar una linea
  repaint();
}
```

Para el evento *MouseRelease* se debe identificar igualmente cual componente esta seleccionado si se trata de un icono, este evento se limita a liberar la variable *selectedIcon* para no seguir cambiando su posición, si se trata de un Conector se debe identificar en qué punto es liberado, si coincide con un conector de otro icono que esté disponible se establecerá una relación entre ambos iconos trazando una línea entre sus conectores involucrados. Esta relación será almacenada en un arreglo donde se guardaran instancias de la Clase *Connections*. Esta clase se construye a partir de los dos conectores involucrados en la relación y servirá posteriormente para trazar todas las relaciones existente sobre el área de trabajo durante la invocación al método *paint(Graphics g)*. Un ejemplo de una relación establecida se puede apreciar en la figura 58 Se muestra a continuación un fragmento de Figura 62 explicando este método.

Durante el evento *MouseClicked* se revisa la posibilidad de, si el click fue sobre un conector que esta seleccionado, eliminar esa relación o si el click fue sobre el cuerpo de un icono, desplegar el menú emergente. El siguiente fragmento de Figura 63 ilustra este método.

Figura 62: Liberación del click del mouse

```
private void formMouseReleased(java.awt.event.MouseEvent evt) {
  Component press = this.findComponentAt(evt.getPoint());
  //Si fue liberado sobre un Conector
  //y ya existia un Conector seleccionado
  if(press instanceof Conector) && selectedConector != null){
    //Se captura el nuevo conector
    Conector releaseConector = (Conector)press;
    //Se marca como seleccionado
    releaseConector.selected = true;
    //Se guarda la relacion en el arreglo connections
    //de clases Connection
    connections.add(
             new Connection(selectedConector, releaseConector));
    //Se libera la variable selectedConector
    selectedConector = null;
  }
  //Si el click se libera sobre un Icono
  if(press instanceof Icon)){
    //Solo se libera la variable selected Icon
    selectedIcon = null;
  }
  //Se repinta el area de trabajo para actualizar los cambios
  repaint();
}
```

Figura 63: Evento Mouse Clicked

```
private void formMouseClicked(java.awt.event.MouseEvent evt) {
  //Se captura el componente seleccionado con el click
  Component press = this.findComponentAt(evt.getPoint());
  //Si fue presionado un Icon
  if(press instanceof Icon){
    selectedIcon = (Icon)press;
    //Si se trata de un click secundario
    if(evt.getButton() == evt.BUTTON2
                        || evt.getButton() == evt.BUTTON3){
      //Despliega el menu emergente
      selectedIcon.getPupMenu().show(
            evt.getComponent(), evt.getX(), evt.getY());
    }
    //Libera la seleccion
    selectedIcon = null;
  } else if(press instanceof Conector){
    selectedConector = (Conector)press;
    //Si ese conector ya esta seleccionado
    if(selectedConector.selected){
      //Se elimina
      this.removeConector(selectedConector);
    }
  }
}
```

La clase *MyCanvas* tiene sobrecargado el método *paint(Graphics g)* lo que permite aprovechar las propiedades de dibujo de la clase que extiende *(JPanel)* así como adicionar nuevas figuras al interactuar sobre la clase *Graphics* asociada a esta clase y que nos provee de diferentes métodos para trazar figuras, escoger colores y repintar los componentes gráficos que están contenidos dentro de la clase y que hayan cambiado de posición.

Durante la implementación de este método primero se hace un llamado al método *paint(Graphics g)* de la clase superior para que haga un redibujado de los componentes que contiene y de esta manera redibujar todos los iconos que posee en sus nuevas posiciones así como los bordes y colores propios de la clase heredada. Posteriormente se recorre el arreglo *connections* que posee todas la conexiones entre lo conectores y los iconos a los que pertenecen trazando líneas que relacionan de manera visual un icono con uno o más de ellos. Al recorrer el arreglo *connections* se extraen de los objetos de la Clase *Connection*. Esta clase consiste de dos instancias de la Clase Conector, identificadas con los nombres *from* y *to*, haciendo alusión a él conector desde donde viene la relación y hacia dónde va. Recordemos que la Clase Conector hereda a la clase *JComponent*, es decir que hereda los métodos *getX()* y *getY()* para calcular su posición dentro del la clase que los contiene, que en este caso es *MyCanvas*. De esta manera podemos calcular las coordenadas de los conectores y trazar las líneas que representan la relación. A continuación se presenta un fragmento de figura 64 que ilustra lo explicado anteriormente.

Figura 64: Conectores y trazos de líneas

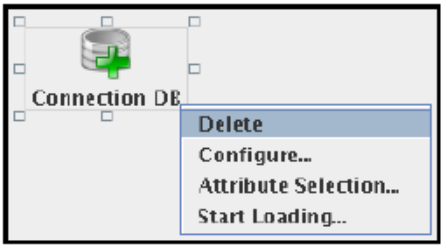
```
public synchronized void paint(Graphics g){
    int  xd, yd, xh, yh;
    super.paint(g);
    Iterator it = connections.iterator();
    while(it.hasNext()){
        Connection aux = (Connection)(it.next());
        xd = aux.from.getX();
        yd = aux.from.getY();
        xh = aux.to.get_X();
        yh = aux.to.get_Y();
        g.setColor(colorEdge);
        g.drawLine(xd, yd, xh, yh);
    }
}
```

5.8.2 Paquete DBConnection

El paquete *DBConnection* se encuentra contenido dentro del paquete *Icons* que a su vez se encuentra dentro del paquete GUI. En este paquete se encuentran las clases necesarias que soportan una conexión a bases de datos a través de un driver JDBC. Hacen parte de este paquete las clases *DBConnectionIcon, connectionWizard, Table, MyCanvasTable, SelectorTable y ScrollableTableModel*.

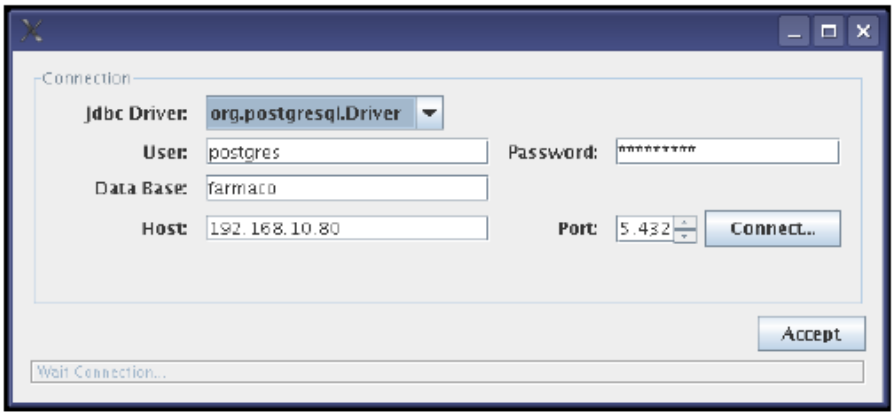
DBConnectionIcon El paquete *DBConnectionIcon* es una clase que extiende a *Icon* del paquete GUI *KnowledgeFlow* y se encarga de proveer un menú desplegable que guie por las etapas de conexión, selección y carga de datos desde una base de datos. La figura 65 muestra la implementación de la clase *DBConnectionIcon* con sus opciones de menú.

Figura 65: Implementación de la clase *DBConnectionIcon*



Al heredar de la clase *Icon*, *DBConnectionIcon* posee por defecto la opción *Delete* en su menú para eliminar este icono del área de trabajo. Al escoger la siguiente opción, Configure, se abre una instancia de la clase *connectionWizard*, la implementación de esta clase se puede ver en la figura 66.

Figura 66: Implementación de la Clase *DBConnectionIcon*



En esta se recoge la información necesaria para establecer una conexión a través del controlador JDBC escogido en el campo JDBC Driver. En este punto hay que aclarar que se hace uso del API SQL de Java, que es un conjunto de clases dispuestas a la interacción con controladores JDBC y al cual se accede importando el paquete *java.sql*. Las clases e interfaces involucradas en la conexión a bases de datos son:

java.sql.DriverManager.

Provee los servicios básicos para el manejo de un conjunto de controladores

JDBC

java.sql.DatabaseMetaData.

Información comprensiva sobre la base de datos en su totalidad.

java.sql.Connection.

Una conexión (sesión) con una base de datos especifica. Se ejecutan las sentencias SQL y los resultados se devuelven dentro del contexto la conexión.

java.sql.Statement.

El objeto usado para ejecutar una sentencia estética SQL y devolver los resultados que esta produce.

java.sql.ResultSet.

Una tabla de los datos que representan un sistema del resultado de la base de datos, que es generado generalmente ejecutando una sentencia SQL que consulta a la base de datos.

java.sql.SQLException.

Una excepción que proporciona la información de un error durante el acceso a bases de datos u otros errores durante la conexión.

A partir del nombre del Driver capturado en la forma se instancia la clase del controlador a través de la instrucción:

Class.forName(JDBCDriver name);

Para el caso de PostgreSQL la instrucción será la siguiente:

Class.forName(“org.postgresql.Driver”);

Con la información referente se construye la url hacia la base de datos y usando la clase *DriverManager* se obtiene una instancia de la *Interface Connection*. El siguiente fragmento de código ilustra esta acción:

```
url = CabeceraDelControlador + “//” + Servidor + “:” + Puerto
        + “/” + NombreDeLaBaseDeDatos;
connection = DriverManager.getConnection(url, usuario, password);
```

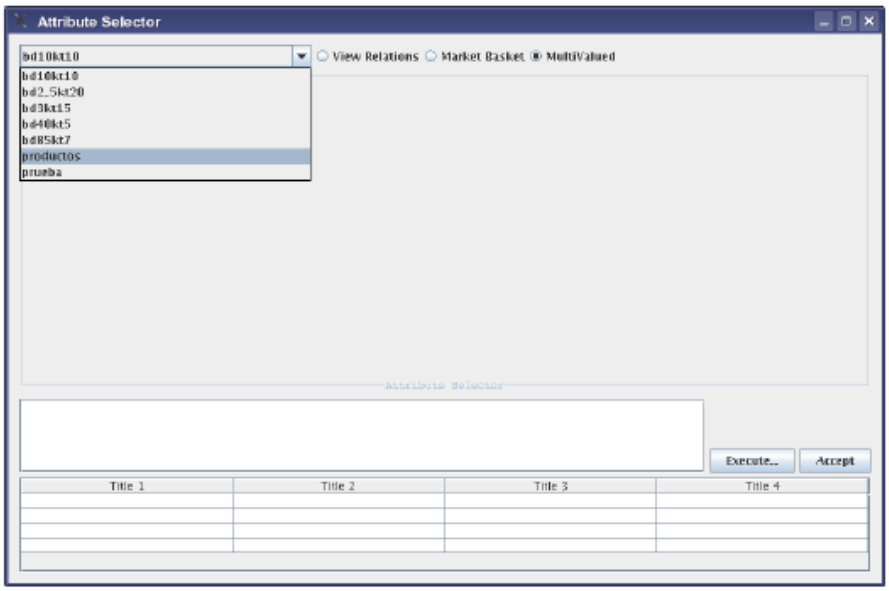
Un ejemplo de la construcción de una conexión a una base de datos PostgreSQL llamada minería a través del usuario ”nceron” con los valores por defecto para el Servidor y el Puerto será la siguiente:

```
url = “jdbc:postgresql://localhost:5432/mineria”;
connection = DriverManager.getConnection(url,”nceron”,”t3o4g2o”);
```

La *interface Connection* obtenida al pulsar el botón Accept de la Clase *connectionWizard* devuelve una instancia de esta conexión al *DBConnectionIcon* que se almacena en la variable connection.

La siguiente opción en el menú desplegable de *DBConnectionIcon* es *Attribute Selection*. Esta alternativa genera un objeto de la clase *SelectorTable*. En la figura 67 se muestra la implementación de esta clase. Esta clase utiliza diferentes objetos del *API Swing* de Java para desplegar y solicitar al usuario información referente al conjunto de datos que quiere minar. Usa un *JComboBox* para listar las tablas que pertenecen a la bases de datos con la cual se estableció conexión, *JRadioButtons* para solicitar al usuario como debe ser considerado el conjunto de datos, como un conjunto multivaluado o bajo la metodología de canasta de mercado, un *JTextArea* para mostrar la sentencia SQL que se está construyendo, un *JTable* que visualizara el contenido de la consulta que se logre construir y una instancia de la clase *MyCanvasTable*, que es una extensión de la clase *JPanel* y es utilizada para desplegar las tablas y relaciones que se establezcan para generar una consulta de manera visual usando la metodología *Drag’n Drop*.

Figura 67: Implementación de la Clase *DBConnectionIcon*



Durante el constructor de la clase se carga el *JComboBox* con los nombres de las tablas que contiene la base de datos conectada. Para obtener esta información se utiliza el método *getTables* de la *interface DatabaseMetaData*. Este método devuelve un *ResultSet* que se recorre para alimentar un Vector que es pasado como parámetro en la construcción del *JComboBox*. El siguiente fragmento de código ilustra en la figura 68 lo anteriormente explicado.

Figura 68: Función getTables

```
  public Vector getTables(){
  ResultSet rs;
  Vector names = new Vector();
  try{
  // Se instancia  DatabaseMetaData a partir de la
  // interfase connection.
  DatabaseMetaData dbmd = connection.getMetaData();
  String[] types = { "TABLE" };
  //Se captura los nombres de la tabla en un ResulSet
  //y se recorre alimentando el Vector names
  rs = dbmd.getTables("%", "%", "%", types);
  while(rs.next()){
    names.addElement(rs.getString(3));
  }
  rs.close();
  }catch(SQLException e){
   e.printStackTrace();
  }finally{
   return names;
  }
}
```

A partir de las tablas que despliegue el *JComboBox* podemos escoger de esa lista la(s) tabla(s) que queremos relacionar en el análisis. La figura 69 muestra esta acción. Estas aparecerán dentro del área del *Attribute* Selector, una instancia de la Clase *MyCanvasTable*, donde podemos interactuar con las tablas de forma grafica a y establecer relaciones al tiempo que construimos la sentencia SQL para consultar la base de datos. Esta sentencia se construye al interior de la Clase *JTextArea* como se puede apreciar en la figura 70.

Figura 69: Tablas de la conexión organizadas en un *JComboBox*

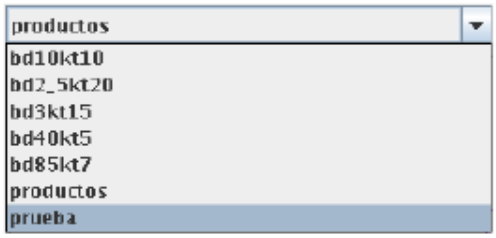
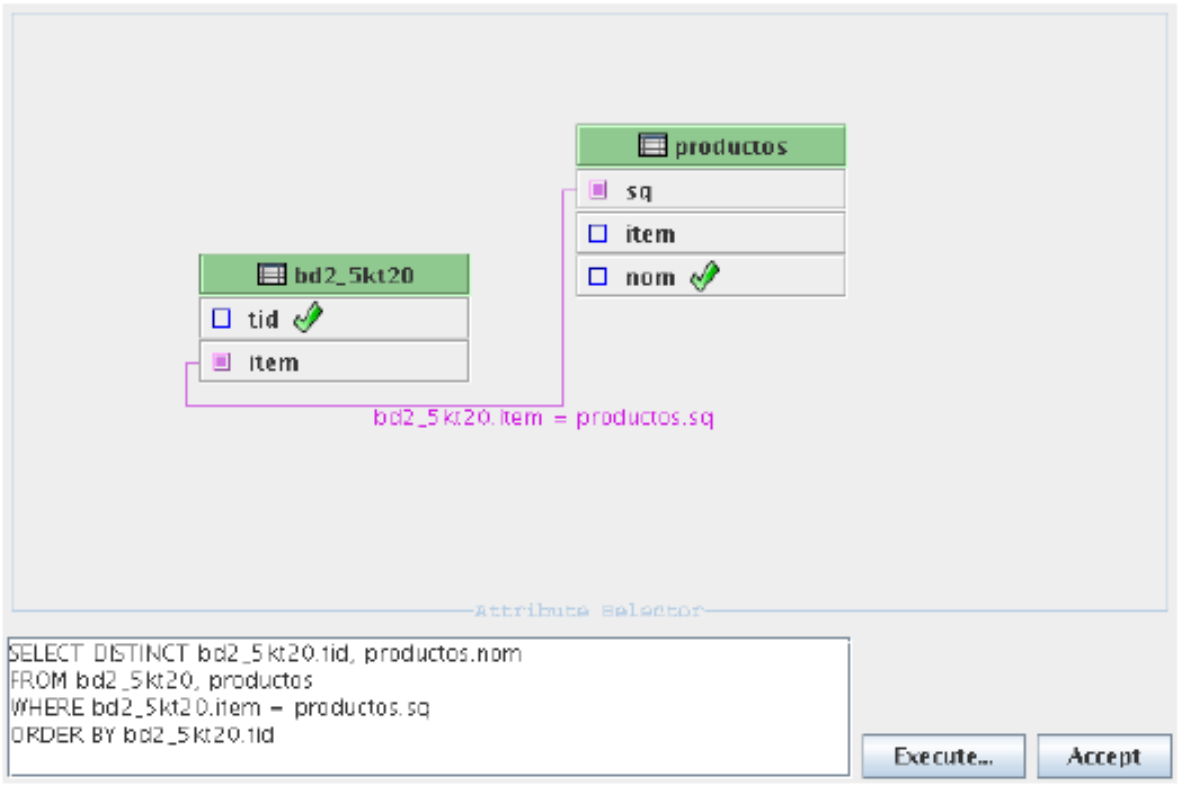


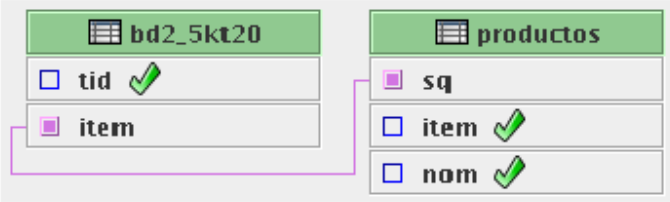
Figura 70: Implementación de la Clase *MyCanvasTable*



Para la interacción de las tablas y las relaciones dentro de *MyCanvasTable* se crearon las Clases *Edge, ConectorTable, Attribute, Table. ConectorTable* es similar a la Clase Conector del paquete GUI *KnowledgeFlow* y es usado para establecer y marcar relaciones entre los atributos de las tablas involucradas. La Clase *Attribute* está conformado por una instancia de *ConectorTable* mas un *JLabel* con el nombre del atributo y la posibilidad de una imagen que indique su selección. La particularidad de la Clase *ConectorTable* es que dependiendo del tipo de atributo el desplegara una imagen diferente como conector si se trata de un atributo de tipo llave primaria, llave foránea o llave mixta. Un ejemplo de esta situación se puede apreciar en la figura 72.

El conjunto de objetos *Attribute* conforman un objeto *Table* que puede estar conformado por un numero *n* de atributos más un *JLabel* que sirva de titulo a la tabla. La implementación final de la Clase *Table* se puede apreciar en la figura 71.

Figura 71: Implementación de la Clase *Table*



La Clase *Edge* cumple la función de guardar los objetos *ConectorTable* involucrados en una relación. De esta manera, la Clase *MyCanvasTable* guarda en un arreglo los objetos *Edges* que representan las relaciones establecidas hasta el momento de manera que al recorrerlo, pueda identificar los conectores involucrados, ubicarlos dentro de la forma y trazar líneas para representar las relaciones entre tablas. Este procedimiento es similar al efectuado en la interfaz principal *(Clase MyCanvas)* con los diferentes tipos de iconos y sus relaciones. La figura 71 ilustra esta implementación.

La Clase *MyCanvasTable* se encarga también de monitorear los eventos del mouse para identificar clicks dentro de las tablas seleccionadas y marcarlos, seleccionar una tabla para su desplazamiento y marcar relaciones entre los atributos de las tablas.

A medida que se marca un atributo de una determinada tabla, este se adicionara al *JTextArea* que construye la sentencia SQL dentro de su campo SELECT y se incluirá el nombre de la tabla de ese atributo dentro del campo FROM, de la misma forma, al establecer una relación se identifican las tablas relacionadas y se incluyen en el campo WHERE de la consulta. Un ejemplo de esta situación se muestra en la figura 72.

Cuando la selección de atributos ha finalizado se puede ver una preliminar de la consulta al pulsar el botón *Execute*. Aquí se instancia un objeto de la Clase *ScrollableTableModel* al cual se pasa como parámetros el atributo *connection* de la clase *MyCanvasTable*, que representa la conexión a la base de datos, junto con una versión textual del contenido del *JTextArea*, que se constituye como la sentencia SQL que queremos consultar.

La Clase *ScrollableTableModel* se encarga de realizar las consultas a la base de datos usando las interfaces *Statament y ResulSet*. La *interface Statement* se crea a partir de la interface *Connection* que llega como parámetro en el constructor de la clase. Esta interface, a través de su método *executeQuery(String query*) dispara una consulta a la base de datos y retorna el resultado en una variable de tipo

ResulSet. El *query* que se pasa como parámetro a este método es el que se construyo como parámetro al constructor de esta clase ver figura 73.

Figura 72: Construcción de la Sentencia SQL

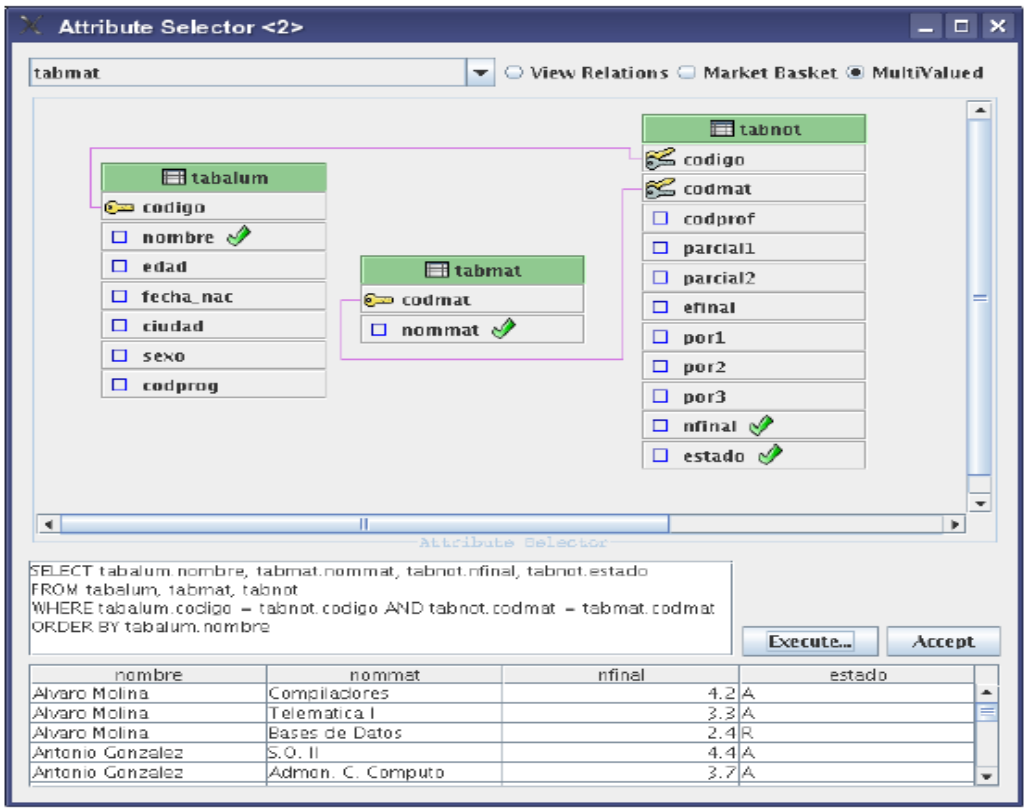


Figura 73: Función executeQuery(String query)

```
Statement statement;
ResultSet resulset;
try {
//Crea una objeto Statement cuyos resultados
//seran sensibles al scroll y de solo lectura
statement = connection.createStatement(
      ResultSet.TYPE_SCROLL_SENSITIVE,
      ResultSet.CONCUR_READ_ONLY);
//Ejecuta la sentencia SQL contenida en query
resultset = statement.executeQuery(query);
} catch (SQLException e) {
throw new SQLException(e.getMessage(), e);
}
```

La Clase *ScrollableTableModel* es la encargada de recorrer el *ResulSet* que contiene el resultado de la consulta y construir un modelo de tabla que es pasado

al *Jtable* de la Clase *MyCanvasTable* para visualizar los resultados de la consulta como se puede apreciar en la figura 74.

Figura 74: Construcción de la previsualización de datos en un JTable

nommat	nombre	nfinal	estado
Compiladores	Alvaro Molina	4.2	A
Telematica I	Alvaro Molina	3.3	A
Bases de Datos	Alvaro Molina	2.4	R
S.O. II	Antonio Gonzalez	4.4	A
Admon. C. Computo	Antonio Gonzalez	3.7	A

Una vez validados los datos que se quiere minar se confirma la aceptación al pulsar el botón *Accept*. En ese momento devolverá la interfaz principal y el último paso para construir el conjunto de datos se realiza a través de la opción *Start Loading* del menú contextual de la clase *DBConnectionIcon* donde, según sea el caso, se invocaría lo métodos *loadMarketBasketDataSet*(), para cargar una instancia del *DataSet* que cubre el modelo de canasta de marcado (Tablas Aniveladas), o *loadMultivaluedDataSet*(), para cargar instancias de *DataSet* de conjuntos multivaluados. Esta instancia de la Clase *DataSet* reposara como atributo de la Clase *DBConnectionIcon* para ser pasada en el momento de una conexión a instancias de las Clases *filterIcon* o *AssociationIcon*.

5.9 ANALISIS, DISEÑO Y MODELADO

En todas las disciplinas de la Ingeniería se hace evidente la importancia de los modelos ya que describen el aspecto y la conducta de "*algo*". Ese "*algo*" puede existir, estar en un estado de desarrollo o estar, todavía, en un estado de planeación. Es en este momento cuando los diseñadores del modelo deben tanto investigar como analizar los requerimientos del producto terminado y dichos requerimientos pueden incluir áreas tales como funcionalidad, *performance* y confiabilidad. Además, a menudo, el modelo es dividido en un número de vistas, cada una de las cuales describe un aspecto especifico del producto o sistema en construcción.

5.9.1 Lenguaje Unificado de Modelado.

(UML, por sus siglas en ingles, *Unified Modeling Language*). El modelado sirve no solamente para los grandes sistemas, aun en aplicaciones de pequeño tamaño se obtienen beneficios de modelado, sin embargo es un hecho que entre más grande

y más complejo es el sistema, mas importante es el papel de que juega el modelado por una simple razón: “El hombre hace modelos de sistemas complejos porque no puede entenderlos en su totalidad”.

Es así como UML es un lenguaje grafico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un “plano” del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables.

El punto importante para notar aquí es que UML es un “lenguaje” para especificar y no un método o un proceso. UML se usa para definir un sistema de software; para detallar los artefactos en el sistema; para documentar y construir, es el lenguaje en el que esta descrito el modelo. UML se puede usar en una gran variedad de formas para soportar una metodología de desarrollo de software, pero no especifica en si mismo que metodología o proceso usar.

UML preescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan.

UML se puede usar para modelar distintos tipos de sistemas: sistemas de software, sistemas de hardware, y organizaciones del mundo real. UML ofrece nueve diagramas en los cuales modelar sistemas.

- ✓ Diagramas de Casos de Uso para modelar los procesos.
- ✓ Diagramas de Secuencia para modelar el paso de mensajes entre objetos.
- ✓ Diagramas de Colaboración para modelar interacciones entre objetos.
- ✓ Diagramas de Estado para modelar el comportamiento de los objetos en el sistema.
- ✓ Diagramas de Actividad para modelar el comportamiento de los Casos de uso, objetos u operaciones.
- ✓ Diagramas de Clases para modelar la estructura estática de las clases en el sistema.
- ✓ Diagramas de Objetos para modelar la estructura estática de los objetos en el sistema.

- ✓ Diagramas de Componentes para modelar componentes.

- ✓ Diagramas de Implementación para modelar la distribución del sistema.

Diagramas de Clases

Las *clases* representan los bloques de construcción más importantes de cualquier sistema orientado a objetos. Una clase es una descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semántica.

Los Diagramas de Clases son utilizados durante el proceso de Análisis y Diseño de los sistemas informáticos donde se crea el diseño conceptual de la información que se manejara en el sistema, los componentes que se encargaran del funcionamiento y la relación entre uno y otro.

Un diagrama de clases presenta las clases del sistema con sus relaciones estructurales y de herencia. La definición de clase incluye definiciones para atributos y operaciones.

Durante el análisis del sistema, el diagrama se desarrolla buscando una solución ideal. Durante el diseño, se usa el mismo diagrama, y se modifica para satisfacer los detalles de las implementaciones.

Un diagrama de clases está compuesto por los siguientes elementos:

- ✓ Clase: atributos, métodos y visibilidad.

- ✓ Relaciones: Herencia, Composición, Agregación, Asociación y Uso.

Diagrama de Paquetes

Cualquier sistema grande se debe dividir en unidades más pequeñas, de modo que las personas puedan trabajar con una cantidad de información limitada, a la vez y de modo que los equipos de trabajo no interfieran con el trabajo de los otros. Un paquete es una parte de un modelo. Cada parte del modelo debe pertenecer a un paquete. Pero para ser funcional, la asignación debe seguir un cierto principio racional, tal como funcionalidad común, implementación relacionada y punto de vista común. UML no impone una regla para componer los paquetes. Los paquetes ofrecen un mecanismo general para la organización de los modelos/subsistemas agrupando elementos de modelado. Cada paquete corresponde a un submodelo (subsistema) del modelo (sistema). Los paquetes son unidades de organización jerárquica de uso general de los modelos de UML.

Pueden ser utilizados para el almacenamiento, el control de acceso, la gestión de la configuración y la construcción de bibliotecas que contengan fragmentos reutilizables del modelo.

Un paquete puede contener otros paquetes, sin límite de anidamiento pero cada elemento pertenece a (está definido en) solo un paquete. El Diagrama de Paquetes, muestra como un sistema está dividido en agrupaciones lógicas mostrando las dependencias entre esas agrupaciones. Dado que normalmente un paquete está pensado como un directorio, los diagramas de paquetes suministran una descomposición de la jerarquía lógica de un sistema.

Los Paquetes están normalmente organizados para maximizar la coherencia interna dentro de cada paquete y minimizar el acoplamiento externo entre los paquetes. Los paquetes son buenos elementos de gestión. Cada paquete puede asignarse a un individuo o a un equipo, y las dependencias entre ellos pueden indicar el orden de desarrollo requerido. Los paquetes se dibujan como rectángulos, las dependencias se muestran como flechas con líneas discontinuas. El operador "::" permite designar una clase definida en un contexto distinto del actual.

Diagrama de casos de uso.

Este diagrama muestra el comportamiento del sistema visto desde la perspectiva del usuario, indica la funcionalidad a cumplir, es una especie de diagrama de comportamiento.

UML define una notación grafica para representar casos de uso llamada modelo de casos de uso, representados por elipses y los actores están representados por las *figuras humanas*. Una caja define los límites del sistema, por ejemplo, los casos de uso se muestran como parte del sistema que está siendo modelado, los actores no.

Existen 3 relaciones principales entre los casos de uso:

Include: En una forma de interacción, un caso de uso dado puede "incluir" otro. El primer caso de uso a menudo depende del resultado del caso de uso incluido. Esto es util para extraer comportamientos verdaderamente comunes desde múltiples casos de uso a una descripción individual. La notación es una flecha rayada desde el caso de uso que lo incluye hasta el caso de uso incluido, con la etiqueta "«include»". Este uso se asemeja a una expansión de una macro donde el comportamiento del caso incluido es colocado dentro del comportamiento del caso de uso base. No hay parámetros o valores de retorno.

Extend: En otra forma de interacción, un caso de uso dado, (la extensión) puede *extender* a otro. Esta relación indica que el comportamiento del caso de uso extensión puede ser insertado en el caso de uso extendido bajo ciertas

condiciones. La notación es una flecha rayada desde el caso de uso extensión al caso de uso extendido, con la etiqueta «extend». Esto puede ser util para lidiar con casos especiales, o para acomodar nuevos requisitos durante el mantenimiento del sistema y su extensión.

Generalization: En la tercera forma de relación entre casos de uso, existe una relación generalización/especialización. Un caso de uso dado puede estar en una forma especializada de un caso de uso existente. La notación es una línea solida terminada en un triangulo dibujado desde el caso de uso especializado al caso de uso general. Esto se asemeja al concepto orientado a objetos de subclases, en la práctica puede ser útil factorizar comportamientos comunes, restricciones al caso de uso general, descríbelos una vez, y enfréntate a los detalles excepcionales en los casos de uso especializados.

Diagrama de Secuencia

Es uno de los diagramas más efectivos para modelar interacción entre objetos en un sistema. Un diagrama de secuencia muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada caso de uso. Mientras que el diagrama de caso de uso permite el modelado de una vista del escenario, el diagrama de secuencia contiene detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario, y mensajes pasados entre los objetos.

Típicamente uno examina la descripción de un caso de uso para determinar que objetos son necesarios para la implementación del escenario. Si tienes modelada la descripción de cada caso de uso como una secuencia de varios pasos, entonces puedes “caminar sobre” esos pasos para descubrir que objetos son necesarios para que se puedan seguir los pasos. Un diagrama de secuencia muestra los objetos que intervienen en el escenario con líneas discontinuas verticales, y los mensajes pasados entre los objetos como vectores horizontales.

Los mensajes se dibujan cronológicamente desde la parte superior del diagrama a la parte inferior; la distribución horizontal de los objetos es arbitraria. Durante el análisis inicial, el modelador típicamente coloca el nombre de un mensaje en la línea del mensaje. Más tarde, durante el diseño, el nombre es reemplazado con el nombre del método que está siendo llamado por un objeto en el otro. El método llamado, o invocado, pertenece a la definición de la case instanciada por el objeto en la recepción final del mensaje.

Diagramas de Colaboración

Diagrama que muestra interacciones organizadas alrededor de los roles. A diferencia de los diagramas de secuencia, los diagramas de colaboración muestran explícitamente las relaciones de los roles. Por otra parte, un diagrama

de colaboración no muestra el tiempo como una dimensión aparte, por lo que resulta necesario etiquetar con números de secuencia tanto la secuencia de mensajes como los hilos concurrentes.

Un diagrama de colaboración es también un diagrama de clases que contiene roles de clasificador y roles de asociación en lugar de solo clasificadores y asociaciones. Los roles de clasificador y los de asociación describen la configuración de los objetos y de los enlaces que pueden ocurrir cuando se ejecuta una instancia de la colaboración. Cuando se instancia una colaboración, los objetos están ligados a los roles de clasificador y los enlaces a los roles de asociación. El rol de asociación puede ser desempeñado por varios tipos de enlaces temporales, tales como argumentos de procedimiento o variables locales del procedimiento. Los símbolos de enlace pueden llevar estereotipos para indicar enlaces temporales.

Un uso de un diagrama de colaboración es mostrar la implementación de una operación. La colaboración muestra los parámetros y las variables locales de la operación, así como asociaciones más permanentes. Cuando se implementa el comportamiento, la secuencia de los mensajes corresponde a la estructura de llamadas anidadas y el paso de señales del programa.

Un diagrama de secuencia muestra secuencias en el tiempo como dimensión geométrica, pero las relaciones son implícitas. Un diagrama de colaboración muestra relaciones entre roles geométricamente y relaciona los mensajes con las relaciones, pero las secuencias temporales están menos claras.

Mensajes: Los mensajes se muestran como flechas etiquetadas unidas a los enlaces. Cada mensaje tiene un número de secuencia, una lista opcional de mensajes precedentes, una condición opcional de guarda, un nombre y una lista de argumentos y un nombre de valor de retorno opcional. El nombre de serie incluye el nombre (opcional) de un hilo. Todos los mensajes del mismo hilo se ordenan secuencialmente. Los mensajes de diversos hilos son concurrentes a menos que haya una dependencia secuencial explicita.

Flujos: Generalmente, un diagrama de colaboración contiene un símbolo para un objeto durante una operación completa. Sin embargo, a veces, un objeto contiene diferentes estados que se deban hacer explícitos. Por ejemplo, un objeto pudo cambiar de localización o sus asociaciones pudieron diferenciarse. Los diferentes símbolos de objeto que representan un objeto se pueden conectar usando flujos “become” o “conversión”. Un flujo “become” es una transición, a partir de un estado de un objeto a otro. Se dibuja como una flecha de línea discontinua con el estereotipo “become” o “conversión” y puede ser etiquetado con un número de serie para mostrar cuando ocurre. Un flujo de conversión también se utiliza para mostrar la migración de un objeto a partir de una localización a otra distinta.

5.9.2 El Proceso Racional Unificado o RUP (Rational Unified Process).

Los orígenes de RUP se remontan al modelo espiral, es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. RUP es en realidad un refinamiento realizado por Rational Software del más genérico Proceso Unificado. Sus principales características son:

- ✓ Forma disciplinada de asignar tareas y responsabilidades (quien hace que, cuando y como)

- ✓ Pretende implementar las mejores prácticas en Ingeniería de Software
  - Desarrollo iterativo
  - Administración de requisitos
  - Uso de arquitectura basada en componentes
  - Control de cambios
  - Modelado visual del software
  - Verificación de la calidad del software

El RUP es un producto de Rational (IBM). Se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso. Incluye artefactos (que son los productos tangibles del proceso como por ejemplo, el modelo de casos de uso, el código fuente, etc.) y roles (papel que desempeña una persona en un determinado momento, una persona puede desempeñar distintos roles a lo largo del proceso).

El RUP divide el proceso de desarrollo en ciclos, teniendo un producto final al final de cada ciclo, cada ciclo se divide en fases que finalizan con un hito donde se debe tomar una decisión importante:

- ✓ Inicio: se hace un plan de fases, se identifican los principales casos de uso y se identifican los riesgos.

- ✓ Elaboración: se hace un plan de proyecto, se completan los casos de uso y se eliminan los riesgos.

- ✓ Construcción: se concentra en la elaboración de un producto totalmente operativo y eficiente y el manual de usuario.

- ✓ Transición: se implementa el producto en el cliente y se entrena a los usuarios. Como consecuencia de esto suelen surgir nuevos requisitos a ser analizados.

- ✓ Fases del RUP:

- Establece oportunidad y alcance
- Identifica las entidades externas o actores con las que se trata
- Identifica los casos de uso.

6. ANALISIS, DISEÑO Y MODELADO DE RASEMUS

En el desarrollo de este proyecto se diseñó e implementó una herramienta DCBD débilmente acoplada con PostgreSQL. Rasemus busca brindar a los usuarios un manejo sencillo de sus datos y le presenta una forma fácil experimentar algunas técnicas de clustering y evaluación y visualización de sus resultados.

Para lograr a cabo este cometido fue necesario el estudio de varias herramientas DCBD y estadísticas existentes en el mercado, para determinar debilidades y fortalezas de las mismas y de esta manera poder desarrollar una herramienta en la que se integraran todas las fortalezas vistas y se eliminaran las dificultades encontradas.

El desarrollo de la herramienta de código abierto bajo software libre y con metodología orientada a objetos hace posible la reutilización de código y permite acoplar funcionalidades con las que da la posibilidad de incluir nuevas características a la herramienta contribuyendo al mejoramiento de la misma. La interfaz gráfica de la misma permite una interacción amigable entre el usuario y la herramienta, facilitando la visualización e interpretación de los resultados.

Para el análisis y diseño de Rasemus se utilizo RUP que es la que más se adecua para el desarrollo de este tipo de herramientas y se utilizo como apoyo la notación UML

A continuación puede apreciarse la descripción formal del desarrollo del proyecto RASEMUS.

6.1. ANALISIS DE RASEMUS

Para el análisis de RASEMUS, se construyen los siguientes artefactos:

- Tabla de funciones o requerimientos.
- Diagramas de casos de uso.

6.1.1 Funciones

Las funciones o requerimientos que se tuvieron en cuenta para el desarrollo de RASEMUS están plasmados en la tabla 54.

Tabla 54: Funciones de RASEMUS

Ref. #	Función	Categoría	Atributo	Detalles y Restricciones	Categoría
R1	Iniciar Herramienta	Evidente	Interfaz	Amigable al usuario	Obligatorio
R1.1	Mostrar interfaz grafica de la herramienta	Evidente	Interfaz	Pestañas desplegable	Obligatorio
R2	Mostrar mensajes de error, respuesta o ayuda cuando sea necesario	Evidente	Interfaz	Cuadros de dialogo o barra de estado	Deseable
R3	Permitir establecimiento de conexión a fuentes de datos	Evidente	Interfaz	De fácil manejo	Obligatoria
R3.1	Permitir conexiones a Bases de Datos	Evidente	Interfaz		Obligatoria
R3.1.1	Permitir selección de atributos	Evidente	Interfaz		Obligatoria
R3.1.2	Cargar el conjunto de datos	Oculto	Información	Establecer relaciones correctas	Obligatorio
R3.2	Permitir conexiones a archivos planos	Evidente	Interfaz		Opcional
R3.2.1	Recorrer archivo y cargar los datos.	Oculto	Información	El archivo debe cumplir con el formato ARFF.	Obligatorio
R4	Permitir la aplicación de filtros	Evidente	Interfaz		Opcional
R4.1	Permitir eliminar datos nulos	Evidente	Interfaz		Opcional
R4.2	Permitir actualizar datos nulos	Evidente	Interfaz		Opcional
R4.3	Permitir seleccionar un conjunto de registros	Evidente	Interfaz		Opcional
R4.4	Permitir reducir el rango de los datos	Evidente	Interfaz		Opcional
R4.5	Permitir codificar los datos	Evidente	Interfaz		Opcional
R4.6	Permitir reemplazar un valor determinado	Evidente	Interfaz		Opcional
R4.7	Permitir discretizar valores continuos	Evidente	Interfaz		Opcional
R4.8	Permitir Estandarizar los valores numéricos	Evidente	Interfaz		Opcional
R4.9	Permitir crear rangos de los datos numéricos.	Evidente	Interfaz		Opcional
R 4.10	Convertir atributos categóricos a numéricos.	Evidente	Interfaz		Opcional
R5	Aplicar técnicas de clustering	Evidente	Interfaz		Obligatorio

R5.1	Proveer algoritmos que cumplan con las técnicas de clustering particional	Evidente	Interfaz		Opcional
R5.1.1	Implementar algoritmo k-means	Evidente	Interfaz		Opcional
R5.1.2	Implementar algoritmo k-prototype	Evidente	Interfaz		Opcional
R5.1.3	Implementar el algoritmo wayCluster	Evidente	Interfaz		Opcional
R5.1.4	Implementar el algoritmo k-histogram	Evidente	Interfaz		Opcional
R5.2	Proveer algoritmos que cumplan con las técnicas de clustering jerárquico	Evidente	Interfaz		Opcional
R5.2.1	Implementar algoritmo de clustering aglomerativo	Evidente	Interfaz		Opcional
R5.3	Proveer algoritmos que cumplan con las técnicas de clustering basado en densidad	Evidente	Interfaz		Opcional
R6	Permitir visualización de resultados	Evidente	Interfaz		Obligatorio
R6.1	ver los representantes de cada clúster	Evidente	Interfaz		Obligatorio
R6.2	Ver la asignación de clúster de cada registro	Evidente	Interfaz		Obligatorio
R6.3	Permitir ver el resultado de la técnica de clustering jerárquico en un dendograma	Evidente	Interfaz		Obligatorio
R7	Permitir crear archivos arff con los clúster generados	Oculto	Información		Opcional

6.1.2 Diagramas de Casos de Uso.

Figura 75: Rasemus

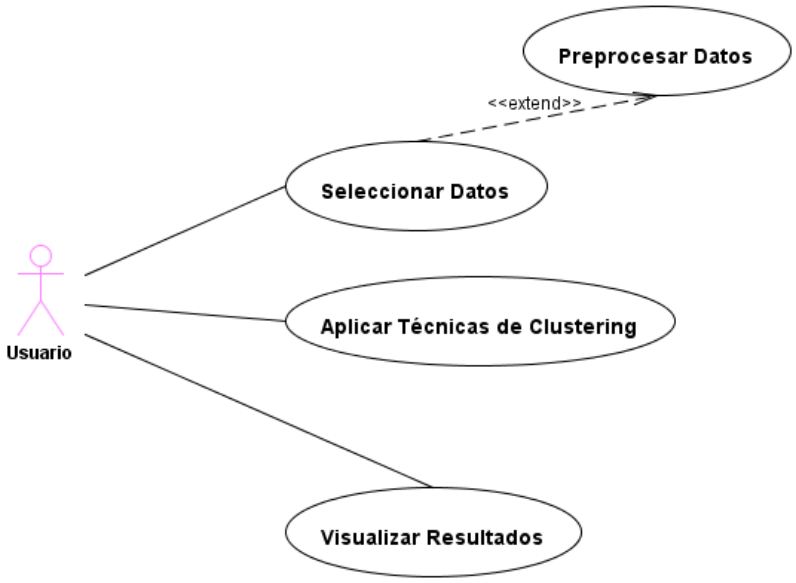


Figura 76: Modulo de Selección.

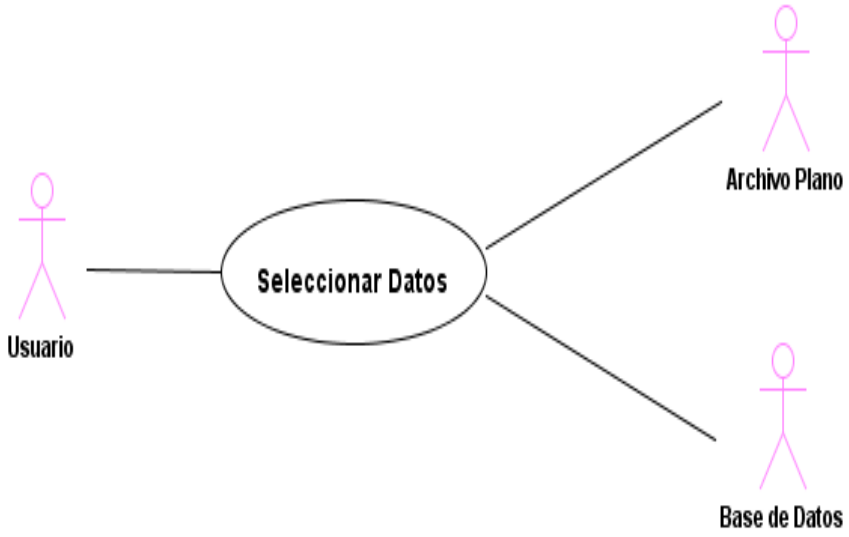


Figura 77: Modulo de Conexión a Base de Datos.

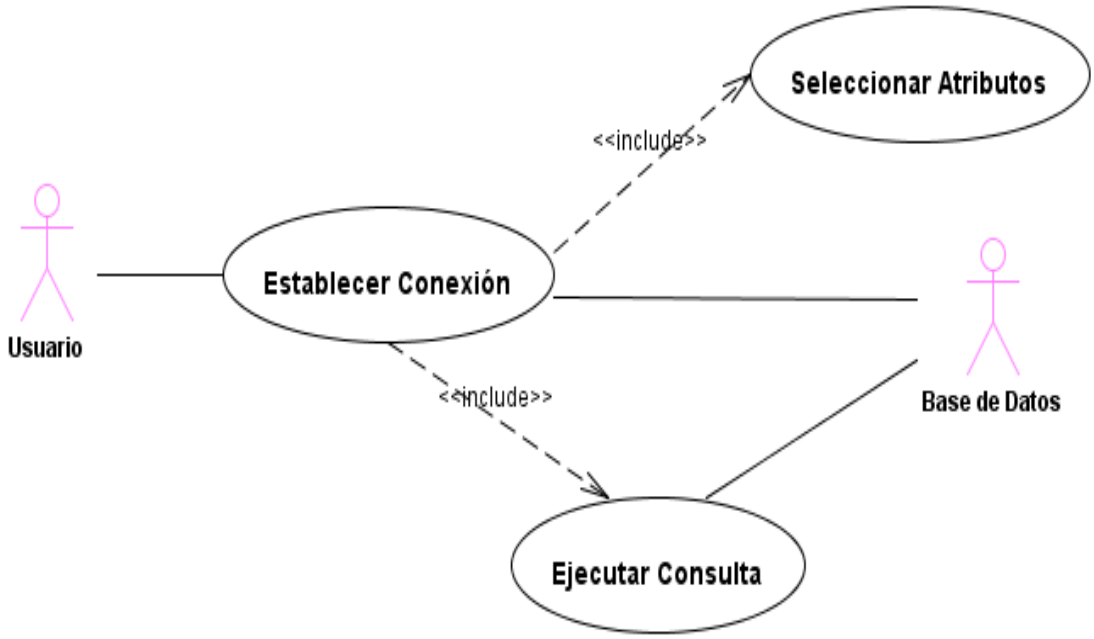


Figura 78: Modulo de Conexión a Archivo Plano.

Figura 79: Modulo de Preprocesamiento.

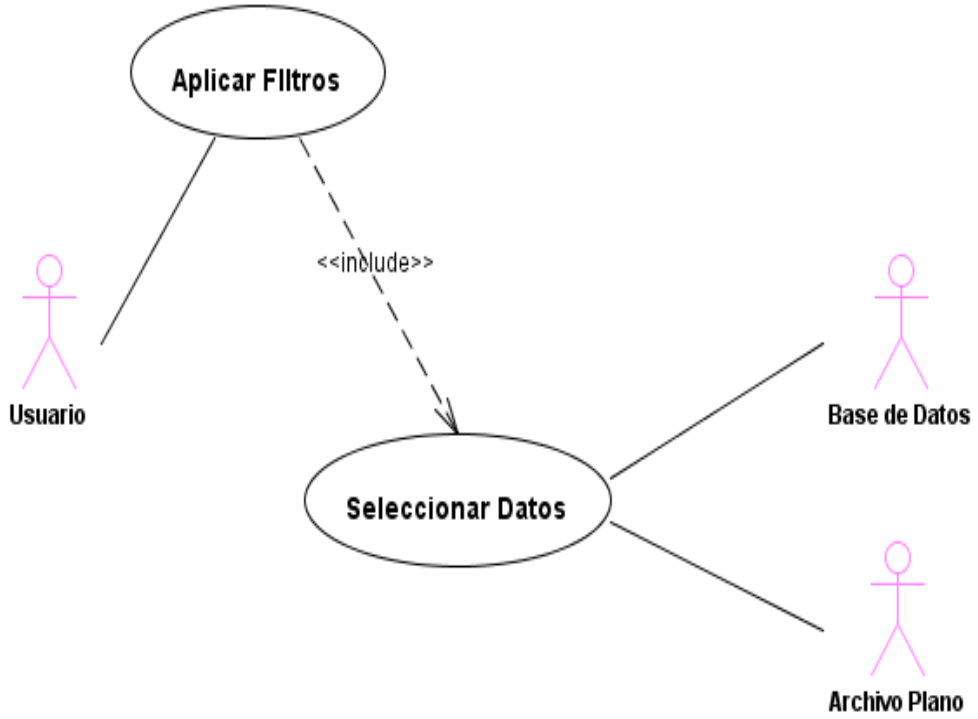


Figura 80: Modulo de Técnicas de Clustering

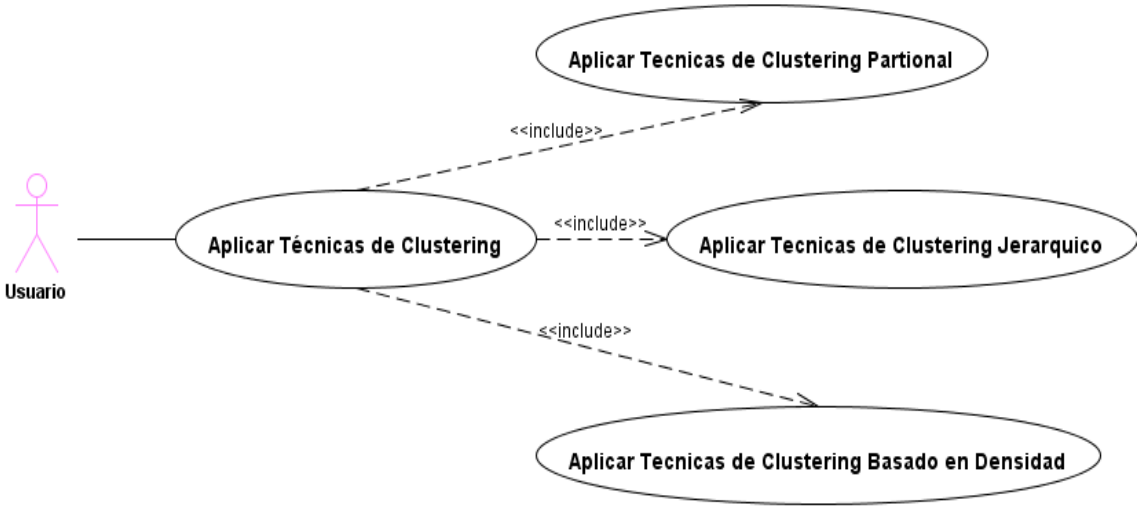
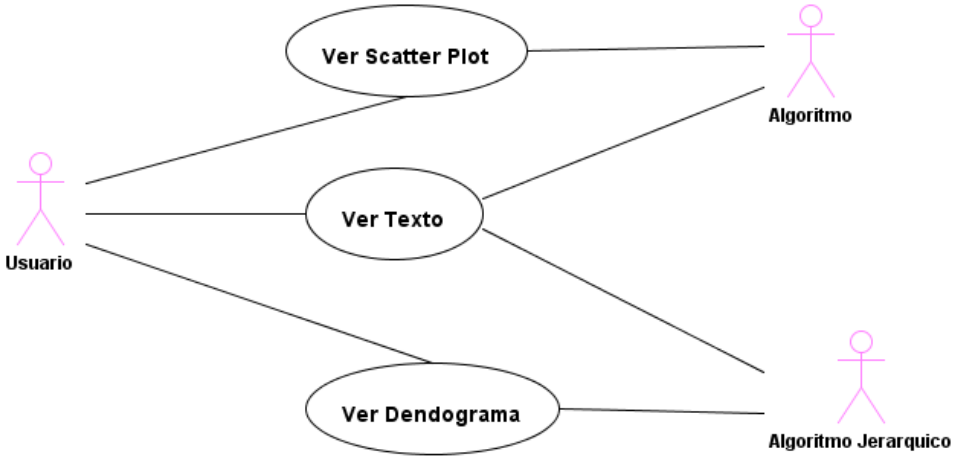


Figura 81: Modulo de Visualización



6.2 DISEÑO DE RASEMUS

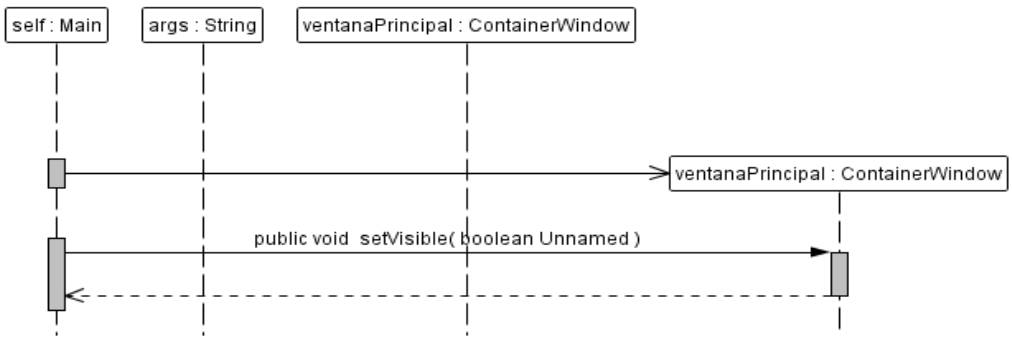
Para el diseño de RASEMUS se hace uso de los siguientes artefactos:

- Diagramas de Secuencia.
- Diagramas de Paquetes.
- Diagrama de Clases

6.2.1 Diagramas de secuencia

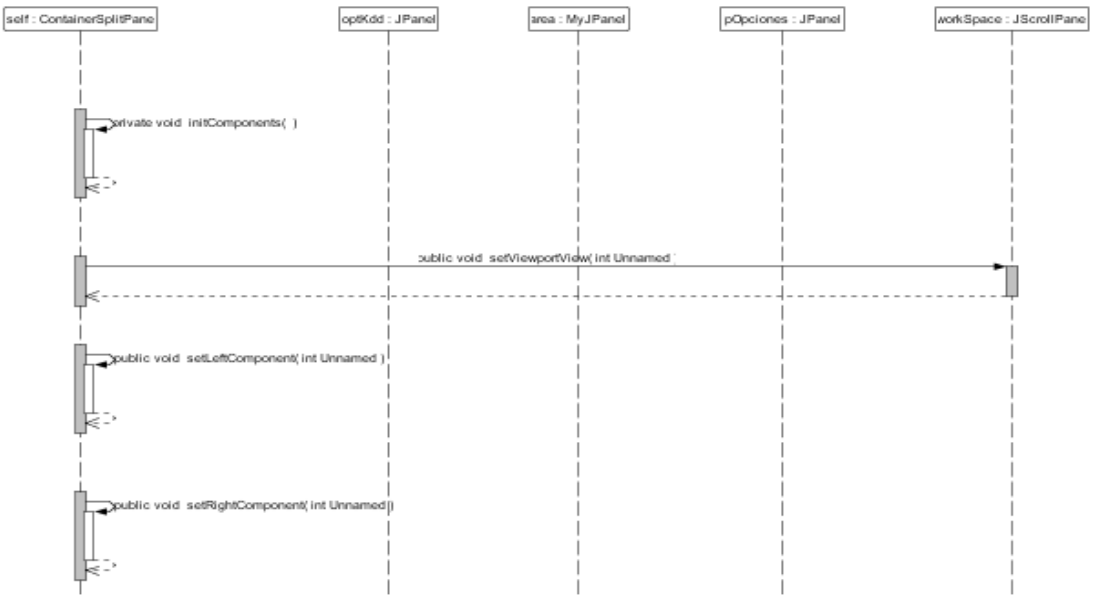
Los diagramas de secuencia de la herramienta se presentan desde la figura 82 hasta la figura 92.

Figura 82: Clase main



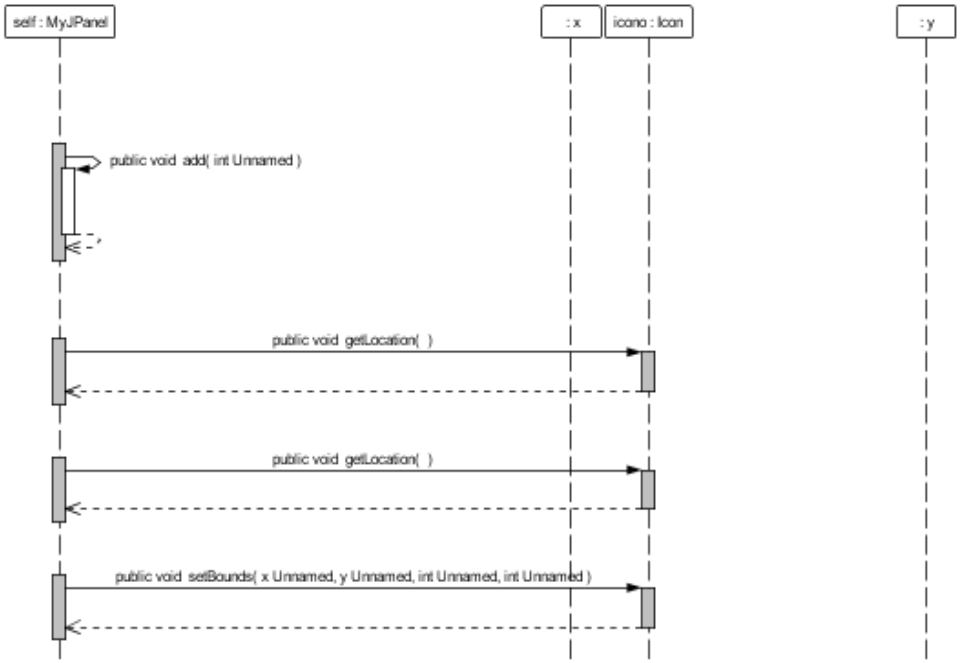
Clase ContainerSplitPane

Figura 83: ContainerSplitPane



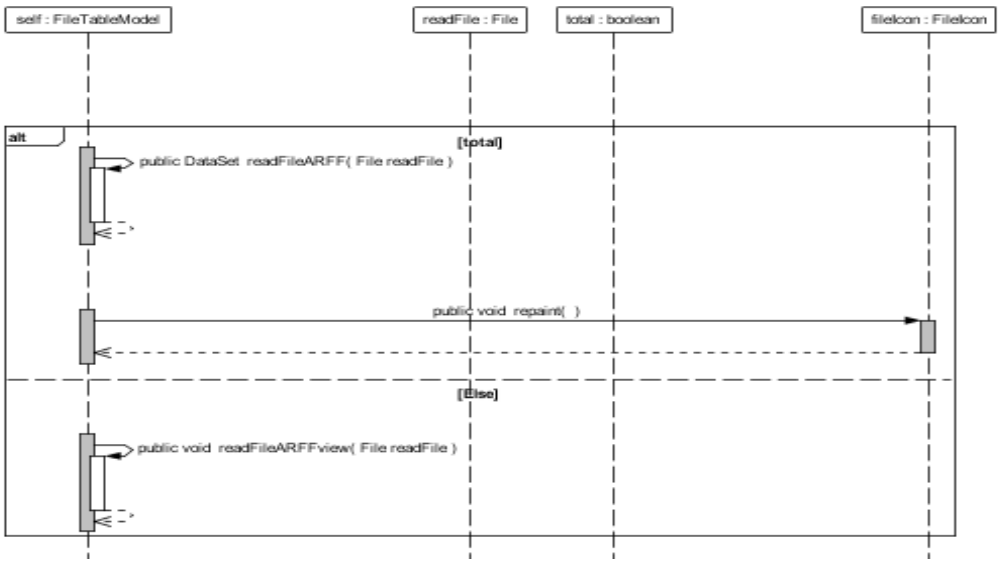
Clase MyJPanel

Figura 84: addIcon clase MyJPanel



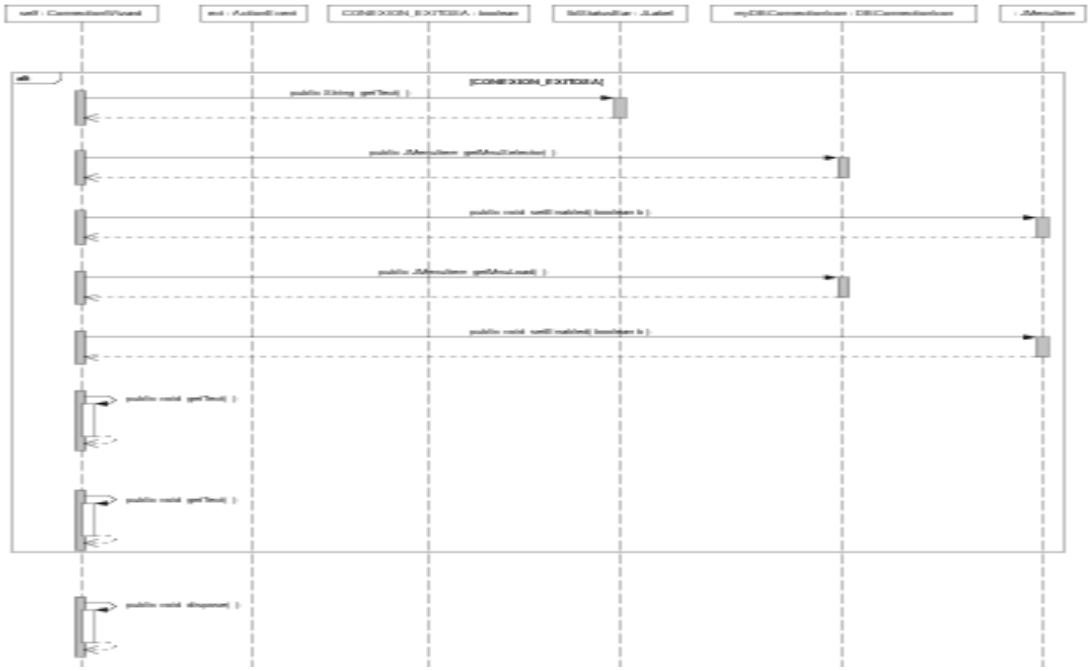
Clase FileTableModel

Figura 85: run clase FileTableModel



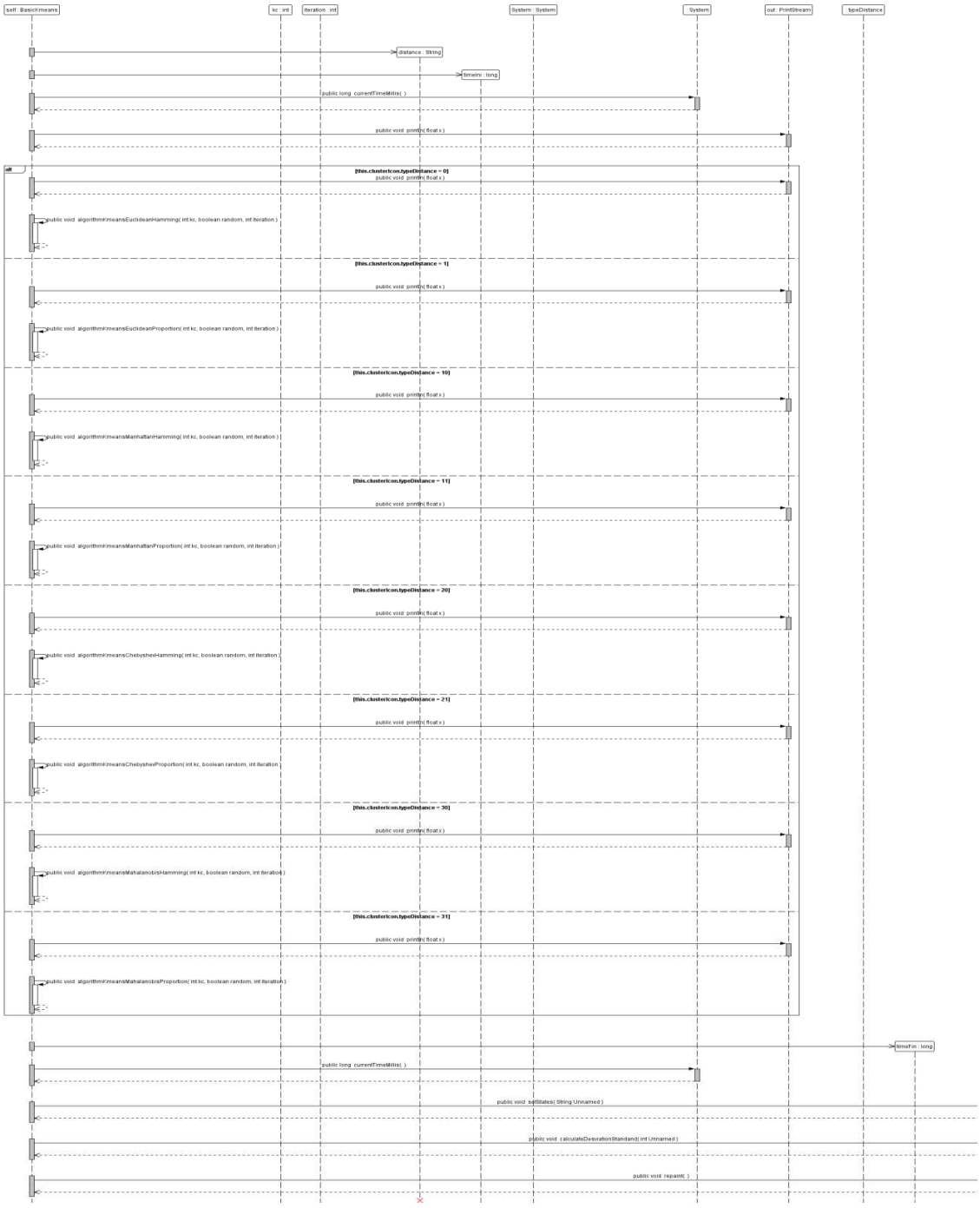
Clase ConnectionWizard

Figura 86: Metodo btnAcceptActionPerformed de la clase ConnectionWizard.



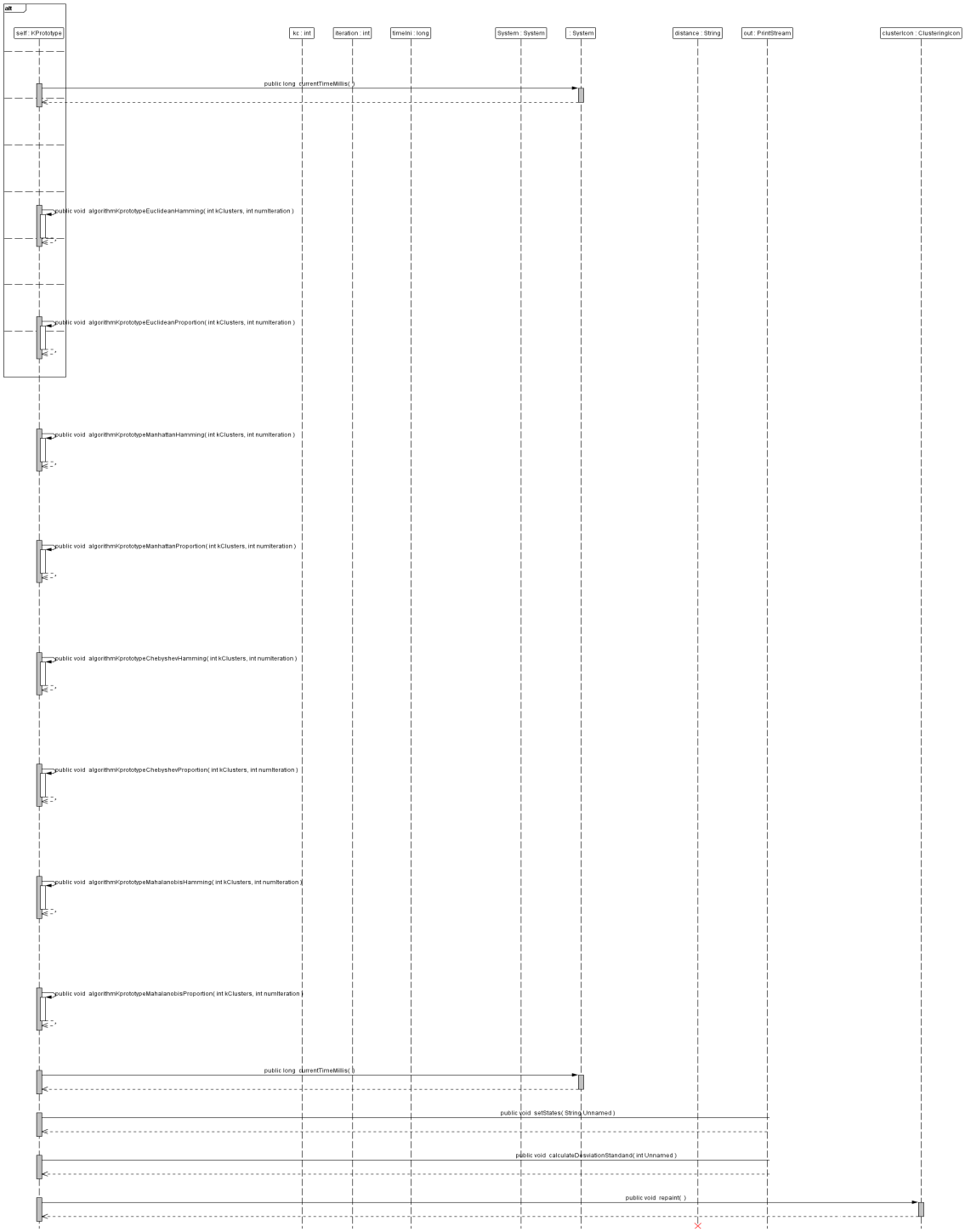
Clase BasicKmeans

Figura 87: run clase BasicKmeans



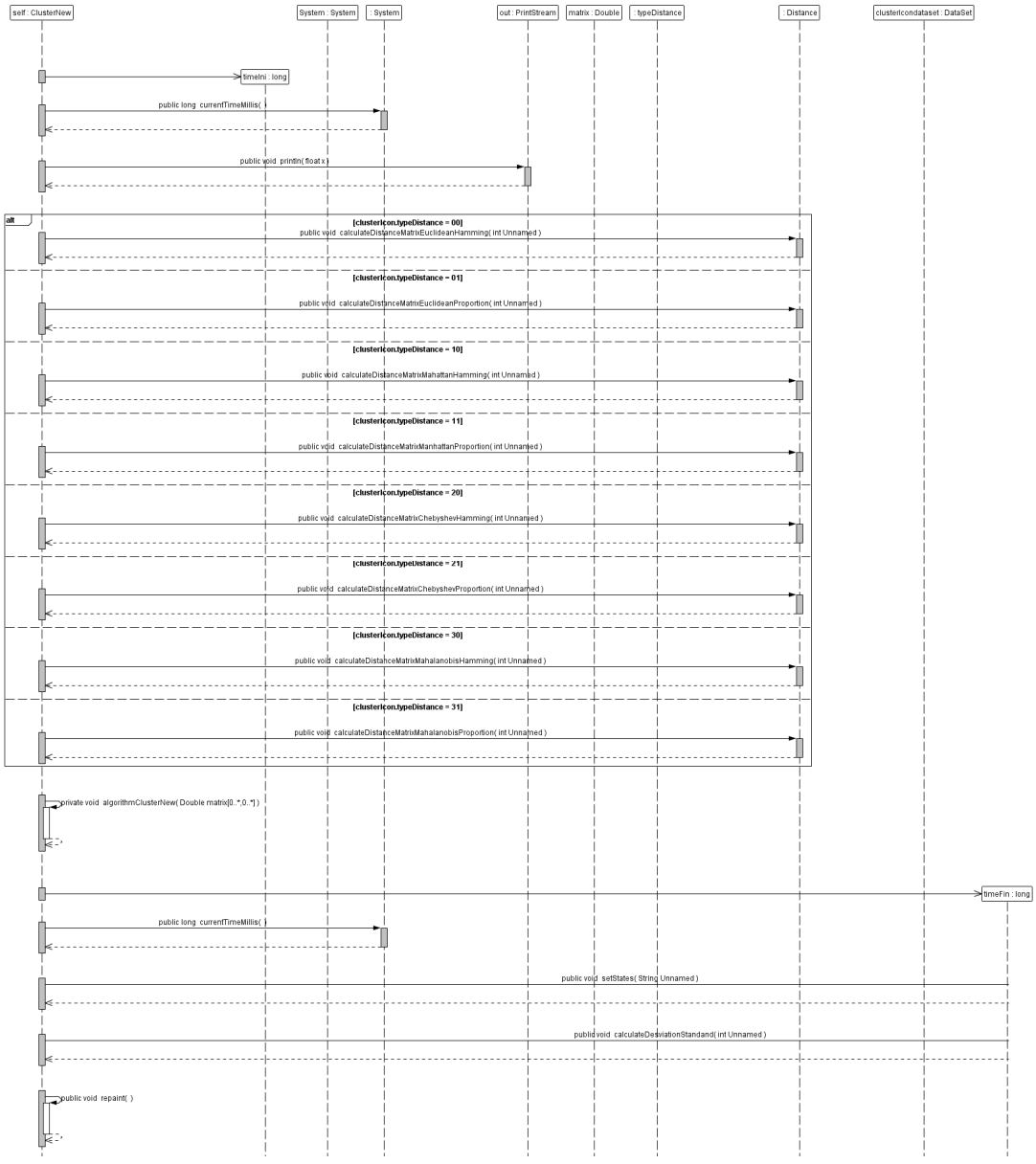
Clase KPrototype

Figura 88: run clase KPrototype



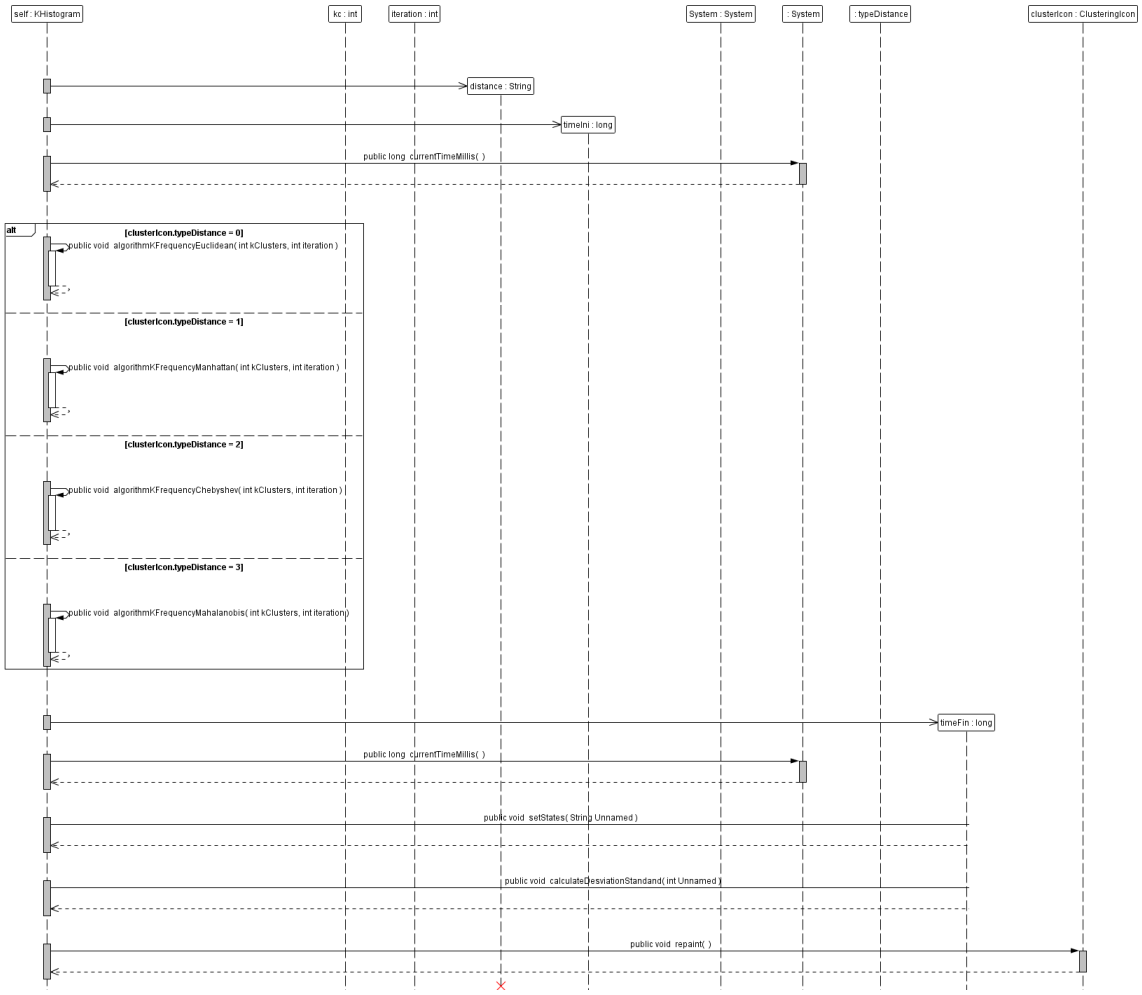
Clase WayCluster

Figura 89: run clase WayCluster



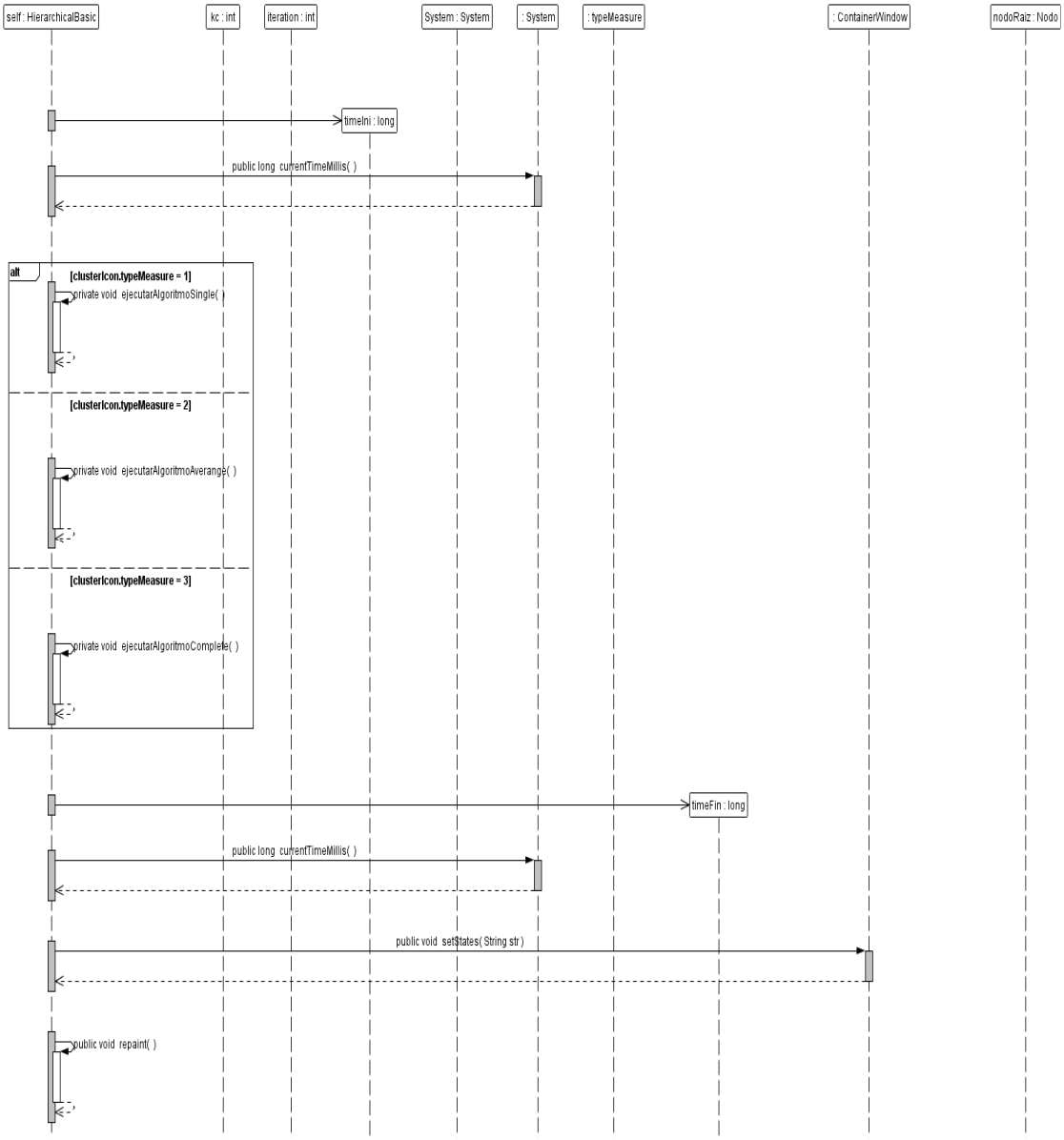
Clase KFrequency

Figura 90: : run clase KFrequency



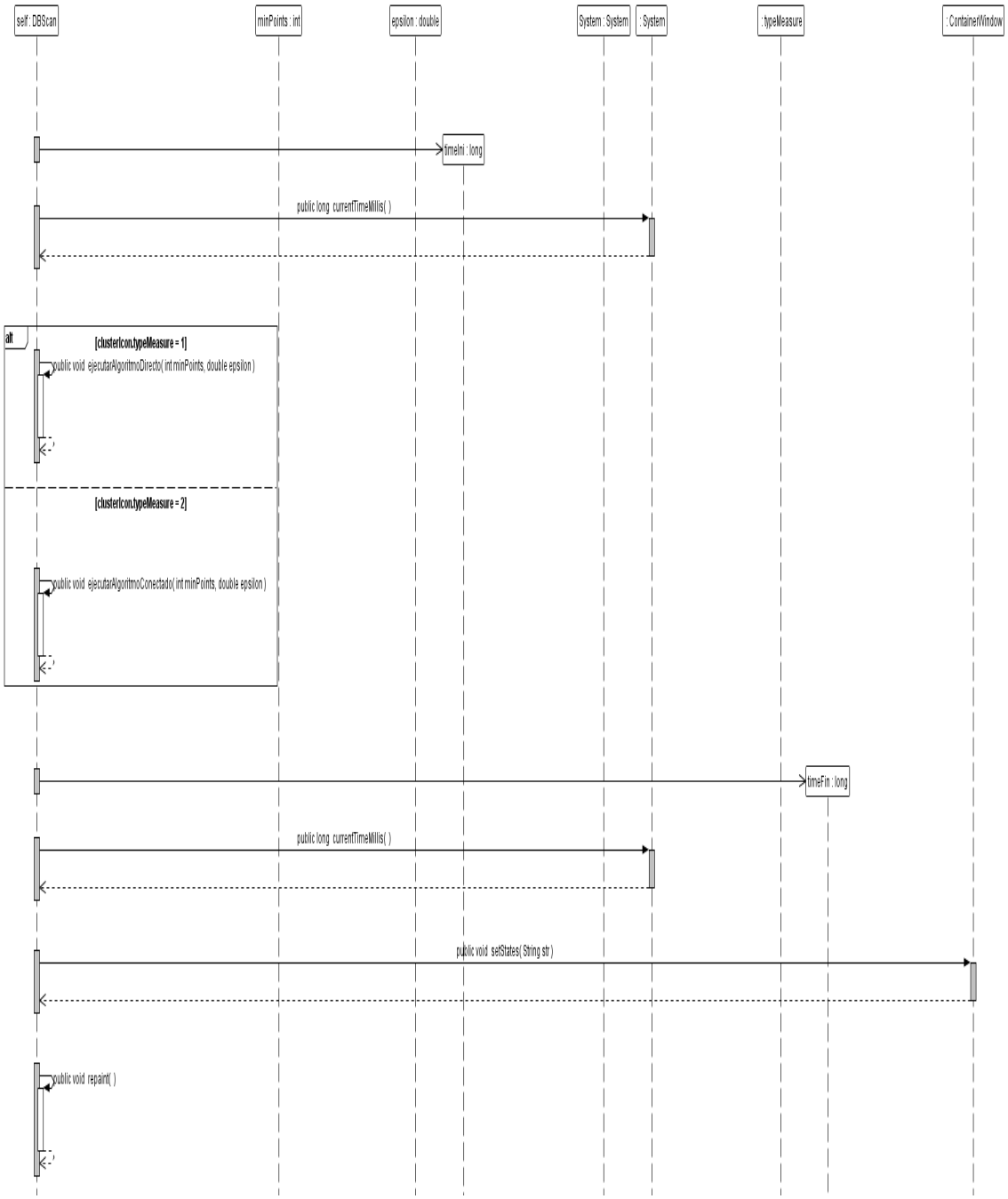
Clase HierarchicalBasic

Figura 91: : run clase HierarchicalBasic



Clase DBScan

Figura 92: Run DBScan



6.2.2 Diagramas de Paquetes

Figura 93: Diagrama de Paquetes Principal.

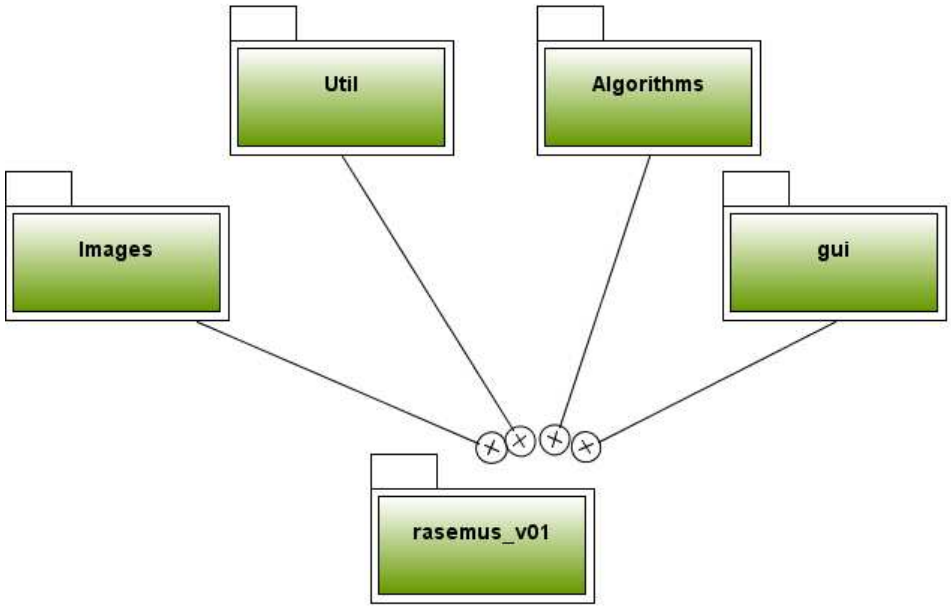


Figura 94: Diagrama de Paquetes Algorithms

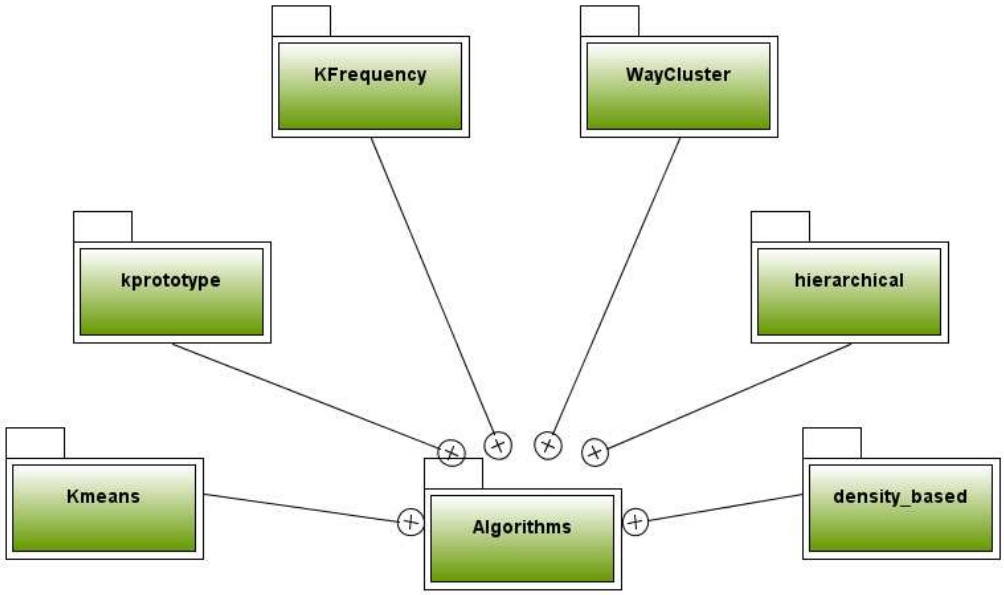


Figura 95: Diagrama de Paquetes Gui.

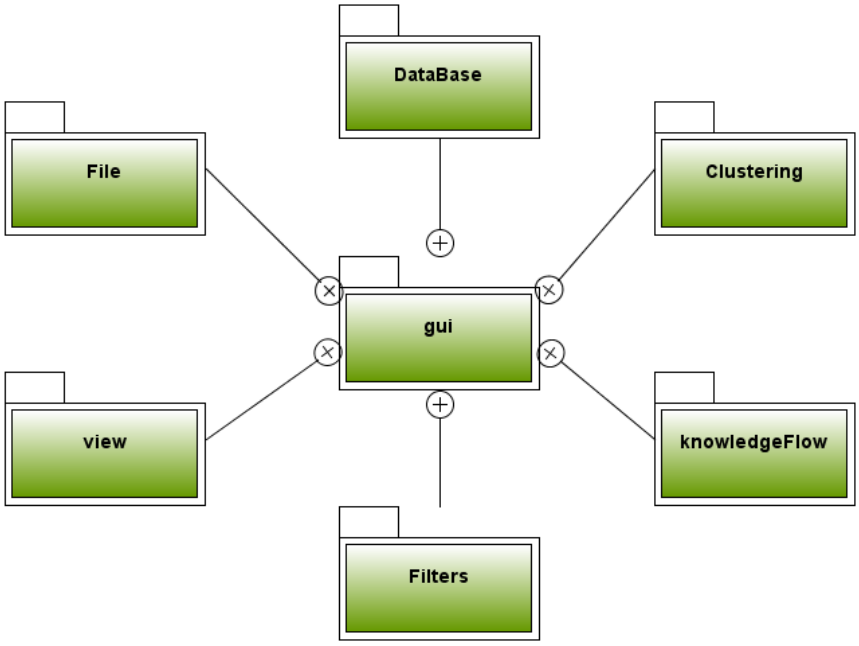
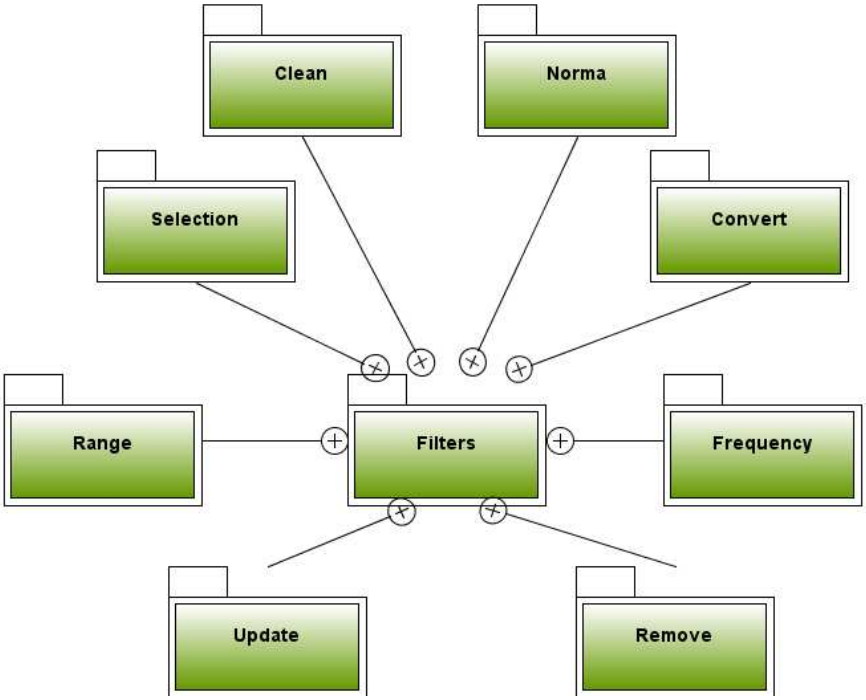


Figura 96: Diagrama de Paquetes Filters.



6.2.3 Diagramas de Clases

Figura 97: Paquete KnowLedgeFlow

Figura 98: Paquete File

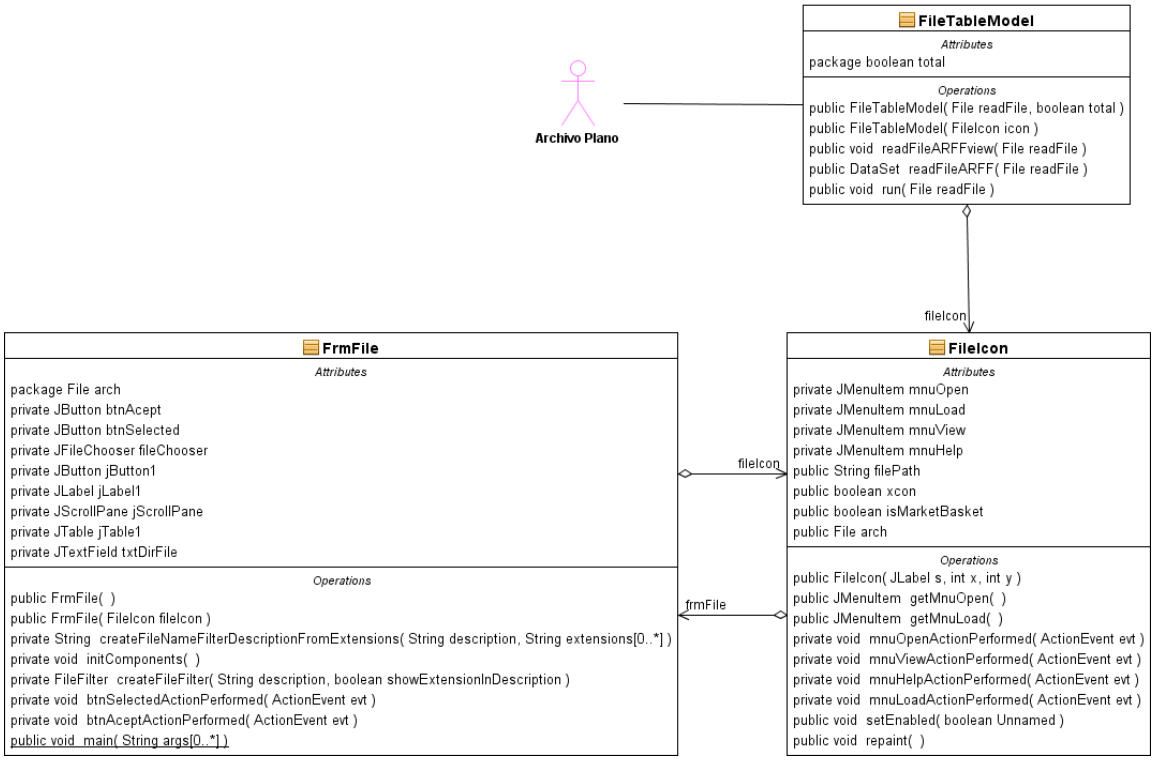


Figura 99: Paquete DataBase

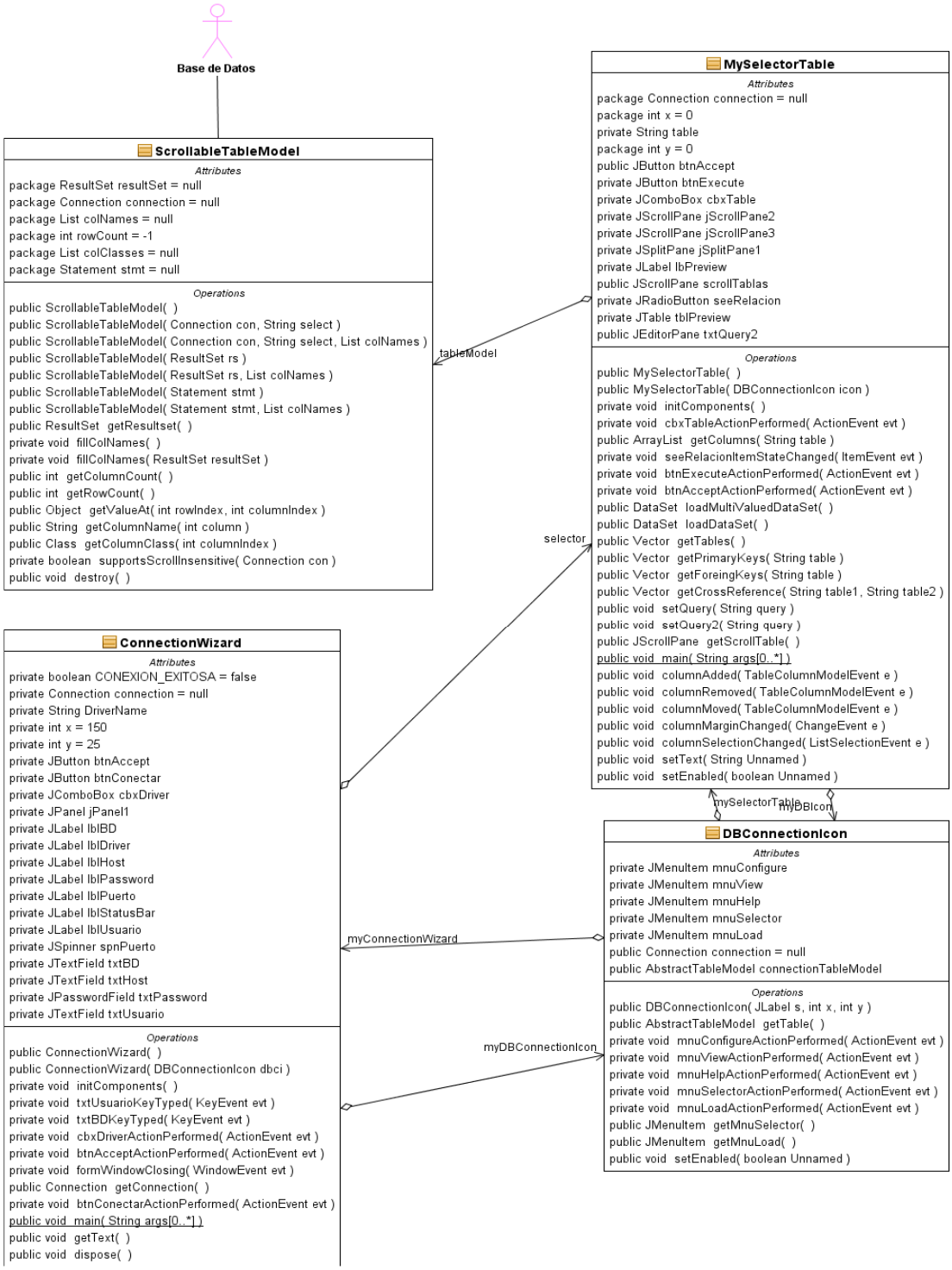
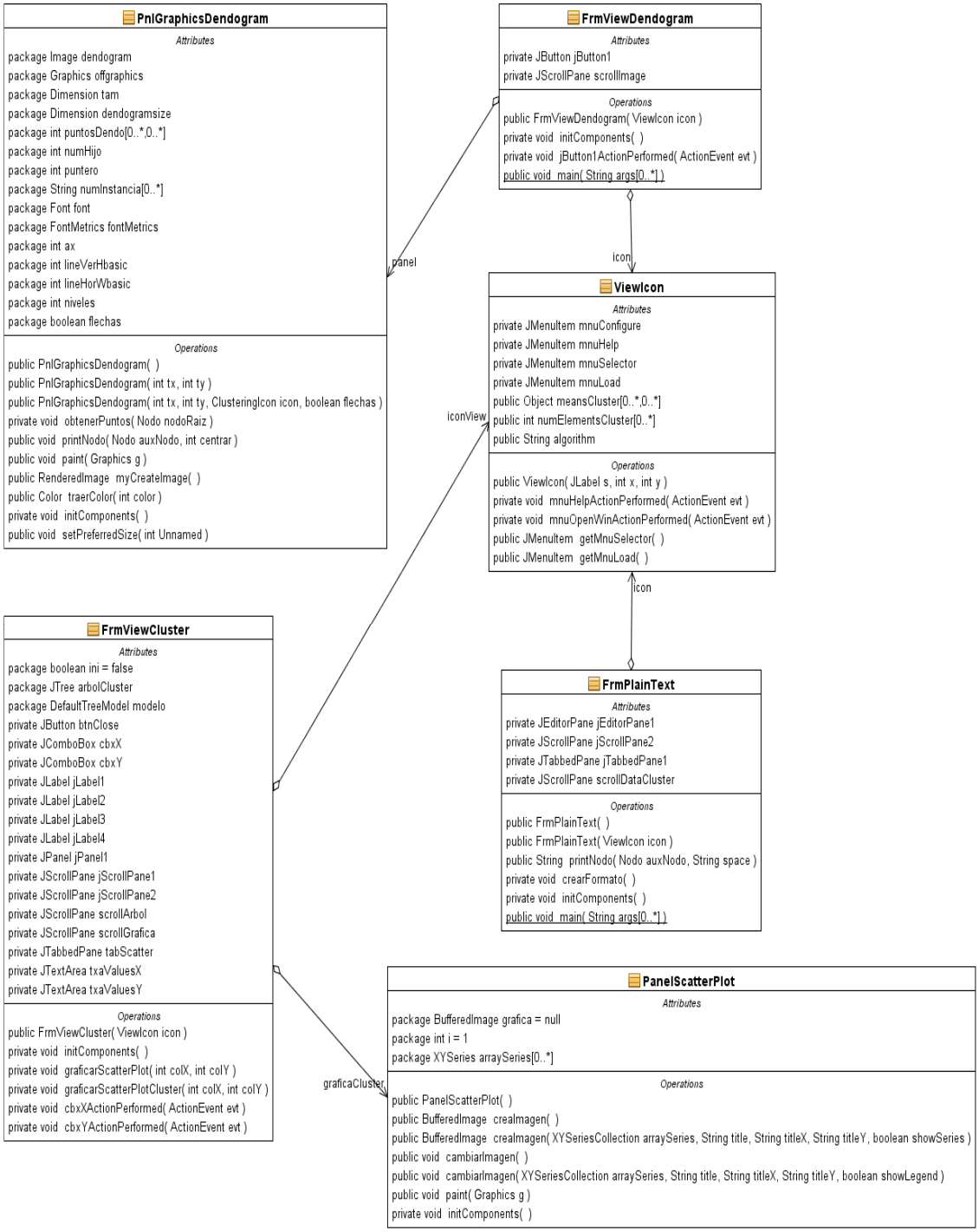


Figura 100: Paquete Clustering

Figura 101: paquete View



7. IMPLEMENTACIÓN RASEMUS

Rasemus se implementa bajo el lenguaje de programación Java™ en su versión 1.6 update 11 con el fin de lograr una herramienta independiente del sistema operativo en el que se trabaje, es decir multiplataforma. Para el desarrollo de Rasemus se utilizo el editor Netbeans en su versión 6.1 y 6.5 y se utilizo la biblioteca grafica de distribución libre JFreeChart, en su versión 1.0.12 con la biblioteca Jcommon en su versión 1.0.15. para realización de graficas. Para la creación de archivos con formatos .pdf y .rtf se utilizo la API iText en su versión 2.1.5. .Para la lectura de archivos con formato.xls se utilizo JExcelApi 2.6.10. para la interfaz grafica y la conexión a bases de datos se reutilizo los paquetes knowLedgeFlow y DBConecction de la herramienta Tariy, a los cuales se les hizo algunas modificaciones, que serán mencionados posteriormente.

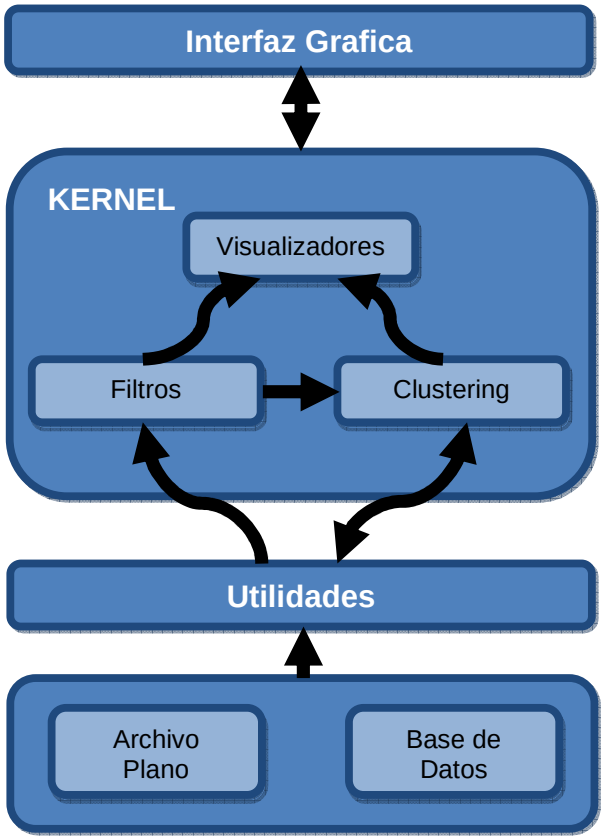
7.1 ARQUITECTURA DE RASEMUS

Rasemus es una herramienta que brindar al usuario diferentes opciones en técnicas de clustering, entre las cuales se puede encontrar técnicas particionales, jerárquicas y basadas en densidad. Implementando los algoritmos básicos de cada técnica, además dentro del proceso DCBD, tiene en cuenta las etapas de selección, preprocesamiento, transformación, minería de datos con las técnicas de clustering y visualización. Para ello se implementaron los módulos de software como se presentan a continuación (ver estructura en la figura 102).

7.1.1 Modulo Utilidades.

En este modulo se encuentran un conjunto de clases que son utilizadas por las demás clases para desarrollar tareas que son comunes, como dar un formato a los datos o permitir el acondicionamiento a algunos objetos gráficos como tablas y demás recursos visuales. También se encarga de establecer la conexión a las diversas fuentes de datos, tanto archivos planos, archivos con formato .xls y archivos con formato .Arff, como a la conexión a bases de datos. También e encustran las clases que permiten exportar reportes en formatos PDF, RTF y HTML. También se encuentran las clases encargada en la creación de archivos con formato .Arff El objetivo de este modulo es agrupar clases que deben estar disponibles para otras clases que las requieran.

Figura 102: Arquitectura de Rasemus.



7.1.2 Kernel de Rasemus

En este modulo reposan los paquetes principales para la ejecución de tareas de descubrimiento de conocimiento. Aquí se encuentran los submódulos de preprocesamiento, clustering y visualizadores.

El módulo de preprocesamiento contiene las diferentes opciones de filtros y rutinas que permiten la adecuación y transformación del conjunto de datos a trabajar, si se cree necesario para lograr mejores resultados con las técnicas de clustering.

El modulo de algoritmos contiene las clases encargadas de aplicar las diferentes técnicas de clustering. En Rasemus se manejan las técnicas de clustering particional, jerárquico y basadas en densidad.

El módulo de visualización contiene las clases necesarias para construir y desplegar estructuras que permiten ver de manera grafica y ágil los resultados de las técnicas de clustering aplicadas.

El objetivo de este modulo es permitir una interacción y comunicación entre los diferentes componentes de los módulos provistos de manera fácil, así como de brindar diferentes perspectivas u opciones para lograr una mejor comprensión del experimento.

7.1.3 Módulo de Interfaz Grafica.

Este módulo da soporte grafico a los todos los demás módulos que lo requieran y se encarga de presentarle al usuario una forma de trabajo amigable y fácil de usar para la construcción de experimentos que involucren la interacción entre los diferentes elementos de la herramienta.

7.2 ESTRUCTURAS DE PAQUETES Y CLASES.

A continuación se describe de manera general la estructura de paquetes de Rasemus y las clases más relevantes en cada paquete. Posteriormente, se detallara algunos aspectos de la forma como se desarrollaron dichas clases.

Paquete util: aquí se encuentran clases utilizadas por la mayoría de elementos de la herramienta que le permiten el desarrollo de sus tareas.

Paquete algorithms: Está compuesto por los paquetes kmeans, kprototype, hierarchical y density_based.

Paquete gui: comprende los paquetes Clustering, DataBase, File, knowledgeFlow, view. Los cuales son los encargados de toda la interfaz grafica de la herramienta.

Paquete Filters: comprende los paquetes Clean, Convert, Frequency, Norma, Range, Remove. Los cuales son los encargados de realizar las operaciones de preprocesamiento y transformación sobre los conjuntos de datos.

7.2.1 Paquete Util.

Del paquete Util las clases más importante son la clase DataSet, la clase Distance y la clase ViewMetaData las cual la se mencionan a continuación, además tiene las clases MiDefaultTableModel y MiJTable que sirven para modificar la presentación de las tablas.

Clase DataSet:

Esta clase contiene el conjunto de datos que el usuario desea trabajar el usuario. Esta clase es requerida por la gran mayoría de elementos de Rasemus, además

de contener los datos, tiene una serie de atributos los cuales contienen características de los datos como el nombre de las columnas, el número de registros o filas, el tipo de datos de las columnas y demás información que permiten un mejor desarrollo de la tareas a realizar. Entre los atributos más importantes de esta clase tenemos:

Name: de tipo String, contiene el nombre del conjunto de datos.

nameColumns: de tipo Object [ ], contiene los nombres de las columnas. Los objetos almacenados son de tipo String.

columnsType: de tipo Object [ ], contiene los tipos de dato de las columnas. Los objetos almacenados son de tipo String

valueCategorical: de tipo Object [ ], contiene vectores un vector que está relacionada con las columnas que son de tipo categórico, almacenando aquí los valores que contenga dicha columna.

numAtributos: de tipo int, número de atributos o columnas del conjunto de datos.

numInstances: de tipo int, número de registros o filas del conjunto de datos.

columnsNumeric de tipo Object[ ], que contiene objeto de tipo Boolean y determina si un atributo es numérico no.

Instancias: de tipo Object[][] son valores del conjunto de datos.

Means: de tipo double [], contiene las medias de los atributos de tipo numérico.

desvMeans: de tipo double [] contiene las desviaciones estandares de los atributos de tipo numérico.

Variance: de tipo double [] contiene las varianzas de los atributos de tipo numérico.

ini de tipo int sirve para establecer desde que atributo se empieza a medir. Es te es importante cuando en el conjunto de datos se tenga como primer columna el nombre o la etiqueta del objeto.

Esta clase tiene dos tipos de constructores un básico DataSet( ); y otro que contiene es Dataset(String name,String [] nameColumns,String[] typeColumns, Object[] valueCategorical,Boolean[] columnsNumeric, Object[][] rows) el cual asigna estos valores a sus respectivos atributos.

Clase Distance:

En esta clase se encuentra todo lo referente a la funciones para calcular distancias entre objetos y funciones para calcular matrices de distancia, como se dijo anteriormente que hay diferentes formulas para medir distancia entre atributos categóricos y diferentes formulas para calcular disimilitud en atributos categóricos. Se implementaron las formulas de distancia, haciendo las siguientes combinaciones representadas en un método estático de la siguiente forma:

- ✓ Distancia Euclidiana y distancia Hamming: Es un método estático que se denomina measureDistanceEuclideanHamming que recibe como parámetros un array de Object[] que son los valores del objeto 1, otro array de objetos Object[] que son los valores del objeto 2 y un tercer array de Objetc Object[], este tipo de array contiene objetos tipo Boolean que son los que van a definir el tipo si es atributo numérico o no y por último se recibe un parámetro tipo int que es el que determina en donde comienza el conjunto de datos. este método retorna un tipo de dato doublé que es resultado de distancia, (Ver figura 103).

Figura 103: Método measureDistanceEuclideanHamming

```
public static double measureDistanceEuclideanHamming(Object[] obj1, Object[]
                                obj2,Object[] columnNumeric,int ini ){
    double distance = 0;
    double valorInterno=0;
    int valorNominal = 0;

    for(int i=ini; i< obj1.length ; i++ ){
       if((Boolean) columnNumeric[i] ){
         valorInterno+=Math.pow( (Double) obj1[i]- (Double) obj2[i] ,2);
       }else{
            if( ( obj1[i] != obj2[i] ))
               valorNominal+=1;
          }

    distance = (Math.sqrt(valorInterno) + valorNominal);

    return distance;
  }
```

- ✓ Distancia Euclidiana y distancia de proporción simple: Es un método estático que se denomina measureDistanceEuclideanProportion que recibe como parámetros un array de Object[] que son los valores del objeto 1, otro array de objetos Object[] que son los valores del objeto 2 y un tercer array de Objetc Object[], este tipo de array contiene objetos tipo Boolean que son los que van a definir el tipo si es atributo numérico o no, un parámetro tipo int que es el que determina en donde comienza el conjunto de datos y por último se recibe un tipo int que es el numero de dato categóricos en el conjunto de elementos a evaluar. este método retorna un tipo de dato doublé que es resultado de distancia, (Ver figura 104).

Figura 104: Método measureDistanceEuclideanProportion

```
public static double measureDistanceEuclideanProportion(Object[] obj1,
        Object[] obj2,Object[] columnNumeric,int ini,int numCategoricos ){

       double distance = 0;
       double valorInterno=0;
       int valorNominal = 0;
        for(int i=ini; i< obj1.length ; i++ ){
          if((Boolean) columnNumeric[i] )
            valorInterno+=Math.pow( (Double) obj1[i]- (Double) obj2[i] ,2);
          else  if(obj1[i].equals(obj2[i]) )
                    valorNominal+=1;
           distance =  Math.sqrt(valorInterno) +  1 -  ((valorNominal)/ (double)(numCategoricos))
         }
       return distance;
    }
```

- ✓ Distancia Manhattan y distancia Hamming: Es un método estático que se denomina measureDistanceManhattanHamming que recibe como parámetros un array de Object[] que son los valores del objeto 1, otro array de objetos Object[] que son los valores del objeto 2 y un tercer array de Objetc Object[], este tipo de array contiene objetos tipo Boolean que son los que van a definir el tipo si es atributo numérico o no y por último se recibe un parámetro tipo int que es el que determina en donde comienza el conjunto de datos. este método retorna un tipo de dato doublé que es resultado de distancia, (Ver figura)..

Figura 105: Método measureDistanceMahattanHamming

```
public static double measureDistanceManhattanHamming(Object[] obj1, Object[]
                                obj2,Object[] columnNumeric,int ini ){
       double distance = 0;
       double valorInterno=0;
       int valorNominal = 0;

       for(int i=ini; i< obj1.length ; i++ ){
          if((Boolean) columnNumeric[i] ){
            valorInterno+=Math.abs( (Double) obj1[i]- (Double) obj2[i] );
          }else{
                if( ( obj1[i] != obj2[i] ))
                   valorNominal+=1;
             }

       distance = (Math.sqrt(valorInterno) + valorNominal);

       return distance;
    }
```

- ✓ Distancia Manhattan y distancia de proporción simple: Es un método estático que se denomina measureDistanceMahattanProportion que recibe como parámetros un array de Object[] que son los valores del objeto 1, otro array de objetos Object[] que son los valores del objeto 2 y un tercer array de Objetc Object[], este tipo de array contiene objetos tipo Boolean que son los que van a definir el tipo si es atributo numérico o no, un parámetro tipo int que es el que determina en donde comienza el conjunto de datos y por último se recibe un tipo int que es el numero de dato categóricos en el conjunto de elementos a evaluar. este método retorna un tipo de dato doublé que es resultado de distancia, (Ver figura 106).

Figura 106: Método measureDistanceMahattanProportion

```
public static double measureDistanceManhattanProportion(Object[] obj1,
        Object[] obj2,Object[] columnNumeric,int ini,int numCategoricos ){

       double distance = 0;
       double valorInterno=0;
       int valorNominal = 0;
        for(int i=ini; i< obj1.length ; i++ ){
          if((Boolean) columnNumeric[i] )
            valorInterno+=Math.abs( (Double) obj1[i]- (Double) obj2[i] );
          else  if(obj1[i].equals(obj2[i]) )
                    valorNominal+=1;
           distance =  Math.sqrt(valorInterno) +  1 -  ((valorNominal)/ (double)(numCategoricos));
         }
       return distance;
    }
```

- ✓ Distancia Chebyshev y distancia Hamming: Es un método estático que se denomina measureDistanceChevyshevHamming que recibe como parámetros un array de Object[] que son los valores del objeto 1, otro array de objetos Object[] que son los valores del objeto 2 y un tercer array de Objetc Object[], este tipo de array contiene objetos tipo Boolean que son los que van a definir el tipo si es atributo numérico o no y por último se recibe un parámetro tipo int que es el que determina en donde comienza el conjunto de datos. este método retorna un tipo de dato doublé que es resultado de distancia, (Ver figura 107).

- ✓ Distancia Chebyshev y distancia de proporción simple: Es un método estático que se denomina measureDistanceChebyshevProportion que recibe como parámetros un array de Object[] que son los valores del objeto 1, otro array de objetos Object[] que son los valores del objeto 2 y un tercer array de Objetc Object[], este tipo de array contiene objetos tipo Boolean que son los que van a definir el tipo si es atributo numérico o no, un parámetro tipo int que es el que determina en donde comienza el conjunto de datos y por último se recibe un

tipo int que es el numero de dato categóricos en el conjunto de elementos a evaluar. este método retorna un tipo de dato doublé que es resultado de distancia, (Ver figura 108).

Figura 107: Método measureDistanceChebyshevHamming

```
public static double measureDistanceChebyshevHamming(Object[] obj1, Object[]
                                obj2,Object[] columnNumeric,int ini ){
      double distance = 0;
      double valorInterno=0;
      int valorNominal = 0;
      double mayor = Double.MIN_VALUE;

      for(int i=ini; i< obj1.length ; i++ ){
         if((Boolean) columnNumeric[i] ){
             valorInterno = (Math.abs((Double)obj1[i] -  (Double) obj2[i]);
            if(mayor < valorInterno)
               mayor = valorInterno;
         }else{
               if( ( obj1[i] != obj2[i] ))
                  valorNominal+=1;
            }

     distance = (Math.sqrt(valorInterno) + valorNominal);

      return distance;
   }
```

Figura 108: Método measureDistanceChebyshevProportion

```
public static double measureDistanceChebyshevHamming(Object[] obj1, Object[]
                                obj2,Object[] columnNumeric,int ini ){
      double distance = 0;
      double valorInterno=0;
      int valorNominal = 0;
      double mayor = Double.MIN_VALUE;

      for(int i=ini; i< obj1.length ; i++ ){
         if((Boolean) columnNumeric[i] ){
             valorInterno = (Math.abs((Double)obj1[i] -  (Double) obj2[i]);
            if(mayor < valorInterno)
               mayor = valorInterno;
         }else{
               if( ( obj1[i] == obj2[i] ))
                  valorNominal+=1;
            }

     distance = (Math.sqrt(valorInterno)   +  1 -  ((valorNominal)/ (double)(numCategoricos)));

      return distance;
   }
```

- ✓ Distancia Mahalanobis y distancia Hamming: Es un método estático que se denomina measureDistanceEuclideanHamming que recibe como parámetros un array de Object[] que son los valores del objeto 1, otro array de objetos Object[] que son los valores del objeto 2 y un tercer array de Objetc Object[], este tipo de array contiene objetos tipo Boolean que son los que van a definir el tipo si es atributo numérico o no , se recibe un parámetro tipo int que es el que determina en donde comienza el conjunto de datos y por último se tiene un array de doublé que es el vector de varianzas del conjunto de datos. este método retorna un tipo de dato doublé que es resultado de distancia , (Ver figura 109).

Figura 109: Método measureDistanceMahalanobisHamming

```
public static double measureDistanceMahalanobisHamming(Object[] obj1, Object[]
                                obj2,Object[] columnNumeric,int ini ){
      double distance = 0;
      double valorInterno=0;
      int valorNominal = 0;

      for(int i=ini; i< obj1.length ; i++ ){
         if((Boolean) columnNumeric[i] ){
           valorInterno+=Math.pow( (Double) obj1[i]- (Double) obj2[i] / variance[i] ,2);
         }else{
               if( ( obj1[i] != obj2[i] ))
                  valorNominal+=1;
           }

      distance = (Math.sqrt(valorInterno) + valorNominal);

      return distance;
   }
```

- ✓ Distancia Euclidiana y distancia de proporción simple: Es un método estático que se denomina measureDistanceEuclideanProportion que recibe como parámetros un array de Object[] que son los valores del objeto 1, otro array de objetos Object[] que son los valores del objeto 2 y un tercer array de Objetc Object[], este tipo de array contiene objetos tipo Boolean que son los que van a definir el tipo si es atributo numérico o no, un parámetro tipo int que es el que determina en donde comienza el conjunto de datos , se recibe un tipo int que es el numero de dato categóricos en el conjunto de elementos a evaluar y por último se tiene un array de doublé que es el vector de varianzas del conjunto de datos. este método retorna un tipo de dato doublé que es resultado de distancia, (Ver figura 110).

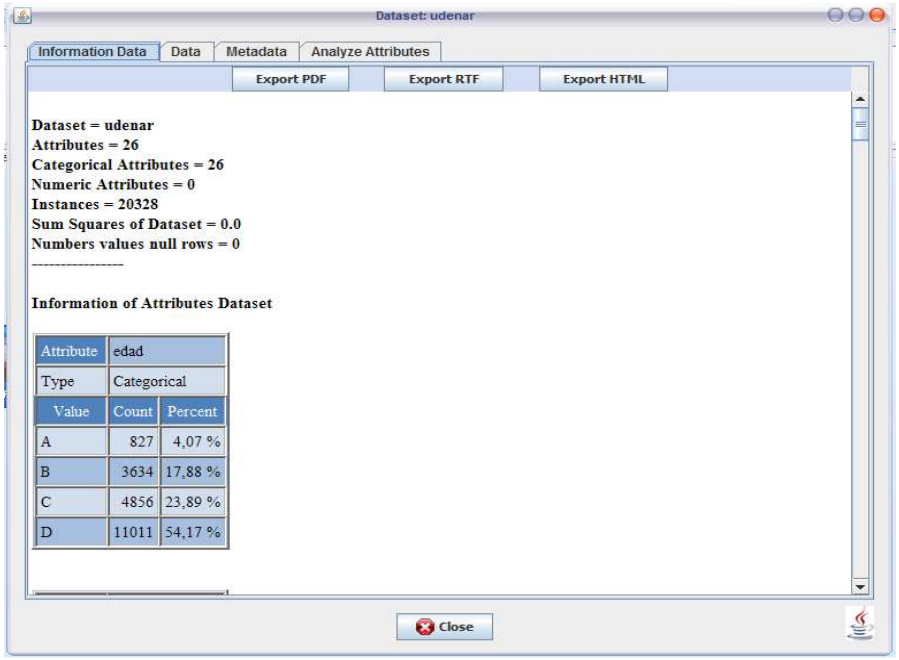
Figura 110: Método measureDistanceMahalanobisProportion.

```
public static double measureDistanceMahalanobisProportion(Object[] obj1,
       Object[] obj2,Object[] columnNumeric,int ini,int numCategoricos ){

      double distance = 0;
      double valorInterno=0;
      int valorNominal = 0;
       for(int i=ini; i< obj1.length ; i++ ){
         if((Boolean) columnNumeric[i] )
           valorInterno+=Math.pow( (Double) obj1[i]- (Double) obj2[i] / variance[i] ,2);
         else  if(obj1[i].equals(obj2[i]) )
                  valorNominal+=1;
           distance =  Math.sqrt(valorInterno) +  1 -  ((valorNominal)/ (double)(numCategoricos))
        }
      return distance;
   }
```

Clase ViewMetaData:

Es una clase que hereda de un JFrame, permite ver información almacenada sobre una instancia de un DataSet, presentando al usuario cuatro pestañas utilizando un JTabbedPane en las cuales se presenta diferentes formas de mostrar la información de los atributos y demás relaciones de un conjunto de datos. La primera pestaña nos información en un formato web, utilizando un JEditorPane configurado para la interpretación de código html (Ver figura 111),

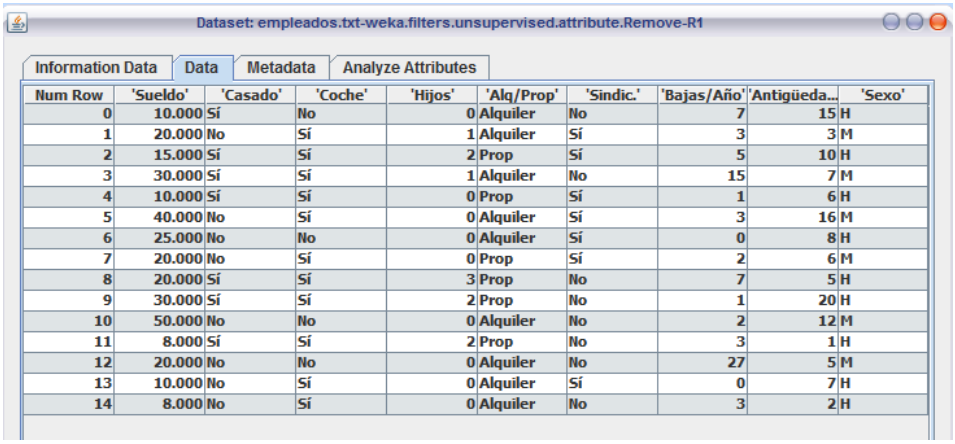
Figura 111: ViewMetaData en la opción Information Data.



En la pestaña “Information data”, se encuentran 3 botones que permiten al usuario crear archivos pdf, rtf, y html. Para ello llaman los eventos de la clase ExportFileFormat, clase la cual utiliza la biblioteca iText.

La otra pestaña denominada *Data* se encuentra los datos encontrados en la instancia de un Dataset, presentados en una instancia de la clase MiJTable (ver figura112),

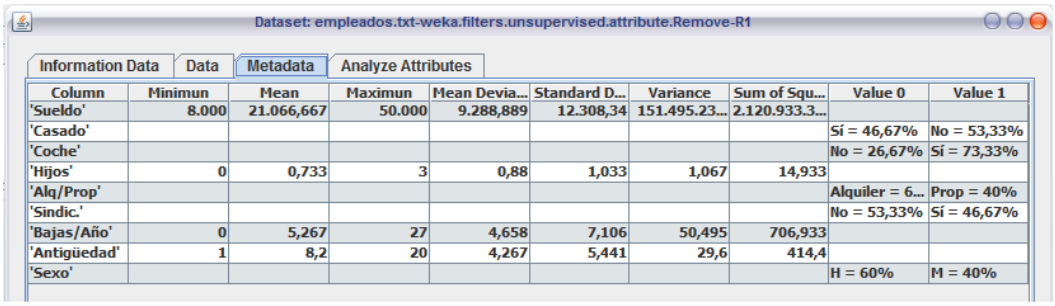
Figura 112: ViewMetaData en la opción Data.



Num Row	'Sueldo'	'Casado'	'Coche'	'Hijos'	'Alq/Prop'	'Sindic.'	'Bajas/Año'	'Antigüeda...'	'Sexo'
0	10.000	Sí	No	0	Alquiler	No	7	15	H
1	20.000	No	Sí	1	Alquiler	Sí	3	3	M
2	15.000	Sí	Sí	2	Prop	Sí	5	10	H
3	30.000	Sí	Sí	1	Alquiler	No	15	7	M
4	10.000	Sí	Sí	0	Prop	Sí	1	6	H
5	40.000	No	Sí	0	Alquiler	Sí	3	16	M
6	25.000	No	No	0	Alquiler	Sí	0	8	H
7	20.000	No	Sí	0	Prop	Sí	2	6	M
8	20.000	Sí	Sí	3	Prop	No	7	5	H
9	30.000	Sí	Sí	2	Prop	No	1	20	H
10	50.000	No	No	0	Alquiler	No	2	12	M
11	8.000	Sí	Sí	2	Prop	No	3	1	H
12	20.000	No	No	0	Alquiler	No	27	5	M
13	10.000	No	Sí	0	Alquiler	Sí	0	7	H
14	8.000	No	Sí	0	Alquiler	No	3	2	H

La siguiente pestaña denominada *Metadata* presenta una instancia de la clase MiJTable en donde se presenta información acerca de los atributos presentes en la clase DataSet, la última es denominada, en la cual los primeras columnas son exclusivas para información de atributos de tipo numérico. Paro los atributos categóricos después de la novena columna se empiezan a adicionar columna columnas con los posibles valores de de dichos atributos (ver figura 113).

Figura 113: ViewMetaData en la opción MetaData.

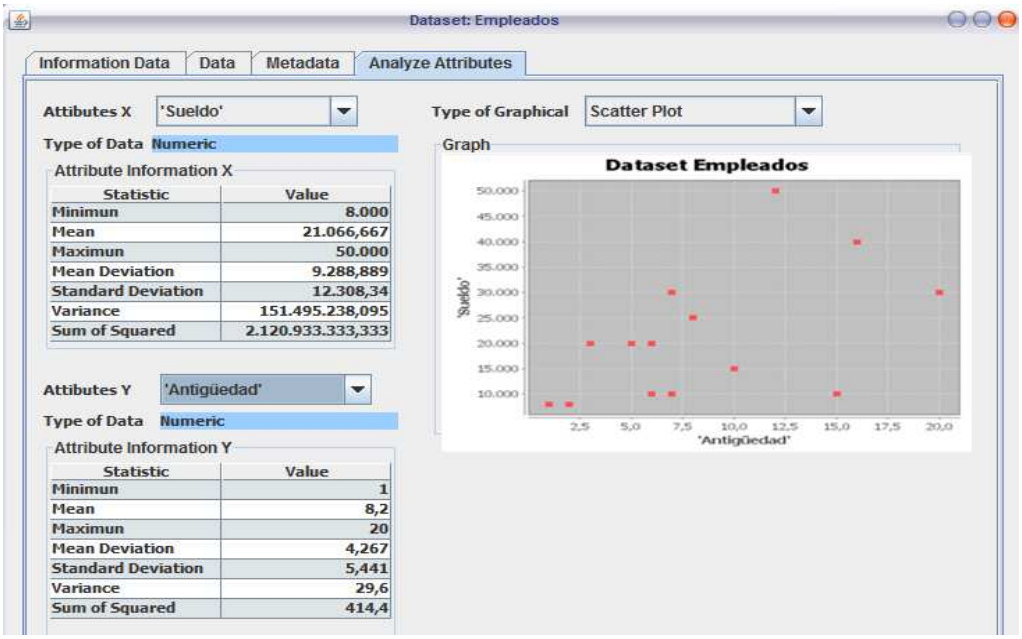


Column	Minimun	Mean	Maximun	Mean Devia...	Standard D...	Variance	Sum of Squ...	Value 0	Value 1
'Sueldo'	8.000	21.066,667	50.000	9.288,889	12.308,34	151.495.23...	2.120.933.3...		
'Casado'								Sí = 46,67%	No = 53,33%
'Coche'								No = 26,67%	Sí = 73,33%
'Hijos'	0	0,733	3	0,88	1,033	1,067	14,933		
'Alq/Prop'								Alquiler = 6...	Prop = 40%
'Sindic.'								No = 53,33%	Sí = 46,67%
'Bajas/Año'	0	5,267	27	4,658	7,106	50,495	706,933		
'Antigüedad'	1	8,2	20	4,267	5,441	29,6	414,4		
'Sexo'								H = 60%	M = 40%

La última opción es *Analisys Attributes* que presenta dos instancias de la clase JComboBox las cuales se cargan los nombres de los atributos encontrados y manejo el evento ActionPerformed, van se va dibujando una grafica generada con

la biblioteca JFreeChat. En un JComboBox están todas las opciones que el usuario tiene para cambiar las diferentes graficas presentadas.

Figura 114: ViewMetaData en la opción Analize Attributes.



7.2.2. Paquete gui.

En este paquete esta toda la interfaz grafica de Rasemus. Las clases que permiten interactuar con el usuario, fueran diseñadas con el editor grafico de netbeans el cual se denomina proyecto Matisse [93]. que permite adicionar elementos como botones, paneles y demás permitiendo su posterior programación.

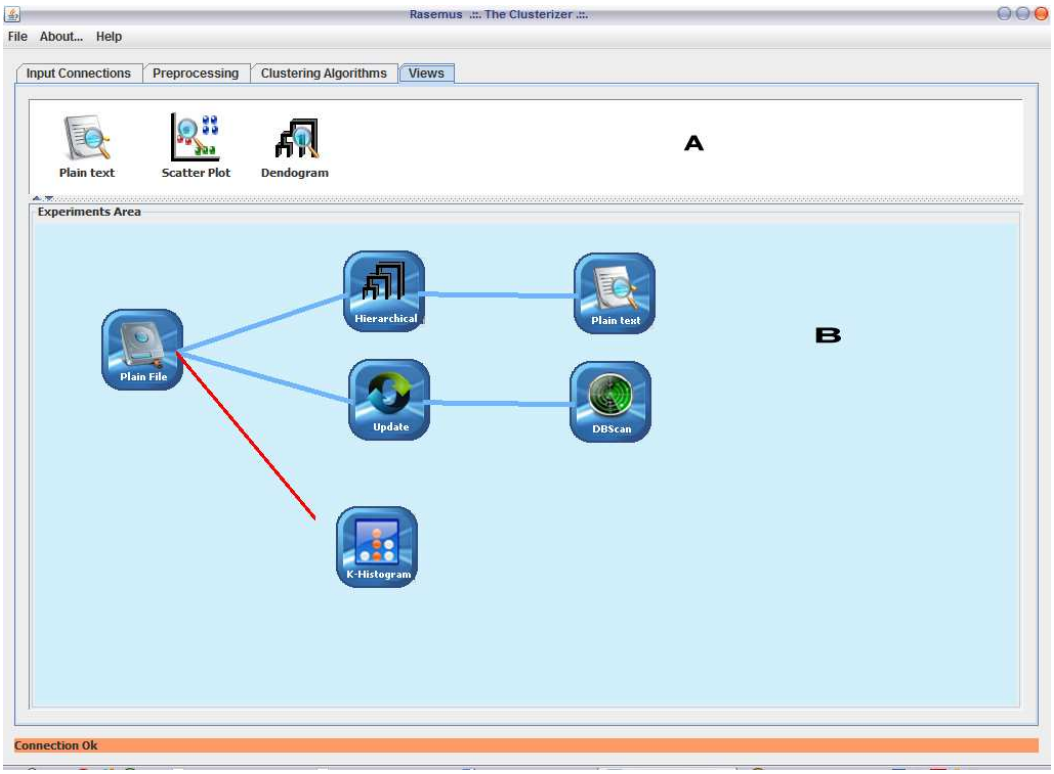
Dentro del paquete gui se puede encontrar los paquetes KnowledgeFlow, File, DataBase, Clustering, View y Filters. El primero contiene todas las clases relacionadas con la interfaz principal y es la que permite trabajar con la técnica Drag and Drop (Arrastrar y Soltar). Los paquetes KnowledgeFlow y DataBase son reutilización de los modulos de TariyKDD denominados KnowledgeFlow y DBConnections los cuales fueron modificados. Los demás paquetes son los encargados de presentarle las tareas que se pueden realizar al usuario y algunas de ellos contienen clases que desarrollan algunas de ellas dentro de los paquetes como veremos a continuación.

Paquete KnowledgeFlow.

En este paquete se encuentran las clases que permiten tener la interfaz de usuario “Darg and Drop” o “Arrastrar y Soltar”. La interfaz principal es la clase

ContainerWindow que hereda de la clase JFrame, que a su vez contiene un JTabbedPane el cual es la encargado de manejar los eventos de cambiar los paneles de las tareas y un ContainerSplitPane que hereda de un JSplitPane que nos permite dividir la ventana en 2 partes, la parte **A** es la de presentación de tareas a través de JPanel, y la parte **B** es el área de trabajo, además de manejar los eventos de adición de tareas al área de trabajo. El área de trabajo es una clase llamada MyJPanel la cual hereda de un JPanel y se encarga del movimiento de las tareas adicionadas, también de realizar las conexiones entre tareas, mantenerlas y de hace las operaciones validaciones de las conexiones ver figura 115.

Figura 115: Interfaz Grafica Rasemus



Las tareas se encuentran dentro de la parte superior de ContaineSplitPane, son paneles que contienen JLabel con el nombre de la tarea y una imagen, cada vez que el usuario da click sobre las pestañas de la clase JTabbedPane se va cambiando los paneles de las tareas, como se ve en la figura 116.

Las tareas que se adicionan al área de trabajo y aparecen como Iconos pertenecen a las clases que eran de la clase Icon la cual hereda de un JPanel. Esta clase es la encargada de manejar los eventos de básicos de un icono como

los eventos, Delete Icon y Name Icon, que borran y cambian el nombre del icono. Los cambios realizados en la clase Icon con respecto a la herramienta Tariy son en el diseño del icono, que se pinta una imagen según al tipo de tarea que pertenezca dicho icono. Para saber qué tipo de icono pertenece la clase Icon tiene como atributo un JLabel el cual nos permite establecer el tipo de tarea a realizar, además la clase Icon contiene 2 instancias de la clase ConnectionArea que son los elementos que nos permiten manejar las líneas de conexión entre iconos, una área está ubicada en el extremo derecho y la otra en el extremo izquierdo. (Ver figura 117).

Figura 116: Paneles de tareas KDD.

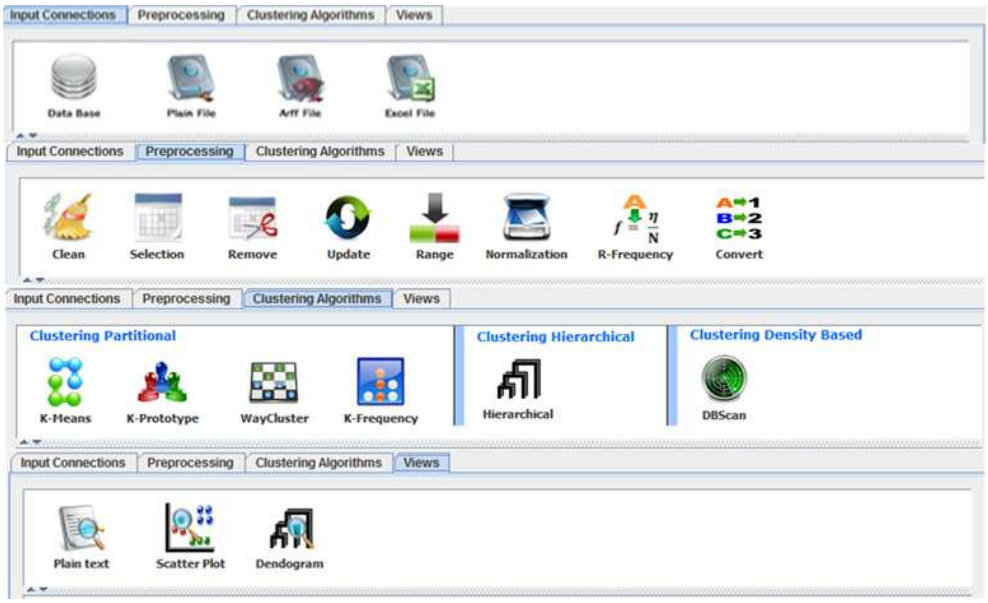
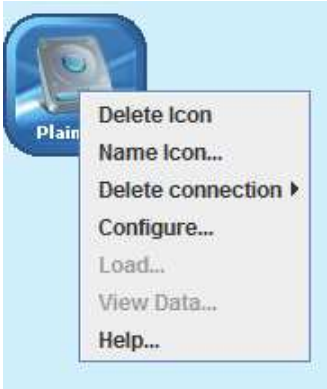


Figura 117: Imágenes de transformación de la clase Icon

Las operaciones que pueden realizar los iconos se presentan al usuario a través de un menú emergente, que es una instancia de la clase JPopupMenu que le permite al usuario de una manera fácil configurar sus tareas y ejecutarlas, (ver figura 118).

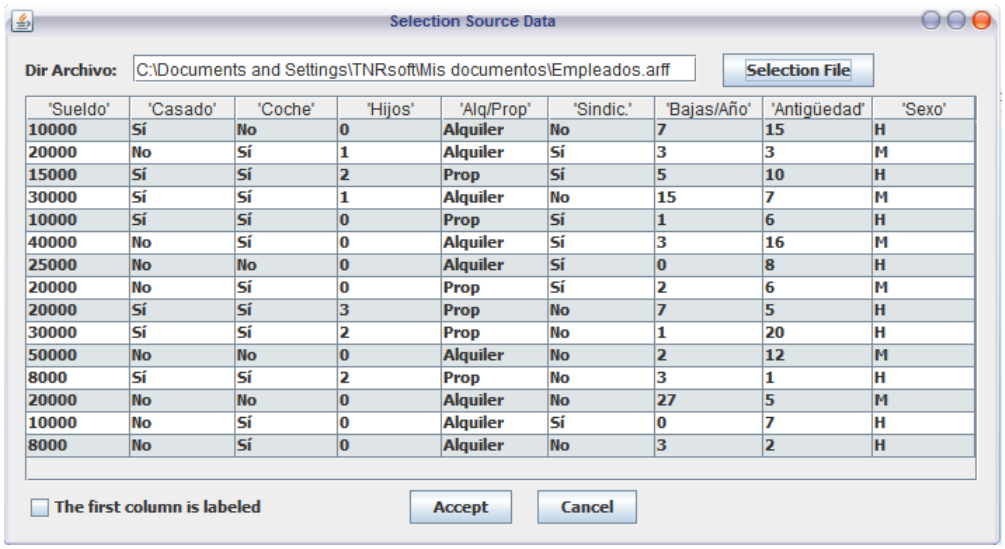
Figura 118: Menú de opciones de los iconos.



Paquete File.

Es paquete es el encargado de permitirle al usuario acceder a la información almacenada en un archivo plano con el formato .arff, .xls o con formato plano. En este paquete encontramos la FileIcon que hereda de la clase Icon. Esta clase brinda al usuario las operaciones que se pueden realizar en cuanto al manejo de archivo, entre las cuales se encuentran *Configure…* que es la operación que le permiten al usuario escoger el archivo que desea trabajar a través de la clase FrmFile que hereda de un JFrame y está compuesta por un JTextField en donde se escribe la ruta del archivo, esta también un JScrollPane que contiene a su vez una instancia de la clase MiJTable en donde se presenta un vista previa de los datos en forma de tabla, también se encuentra el botón *Selection File,* que nos presenta un JFileChooser que nos permite navegar por los medios de almacenamiento, para facilitar la selección del archivo. Ver figura 119.

Figura 119: Instancia de la clase FrmFile.



En el paquete File también encontramos las clases FileTableModelArff, FileTableModelExcel, FileTableModelPlain son las encargadas de establecer la conexión del archivo, leer la información contenida y establecer si tiene el formato requerido o no. Estas clases son accedidas por la clase FrmFile para enviarle el modelo de la tabla para que pueda mostrar la tabla de vista previa de los 100 primeros registros de datos.

La operación que continua en la FileIcon es la operación *load…* que nos permite cargar toda la información del archivo elegido por el usuario, a través de las clases anteiormente mencionadas con su evento readFile que nos devuelve una instancia de la clase DataSet. La última operación que contiene la clase FileIcon es *View Data* nos muestra una instancia de la clase ViewMetaData antes mencionada.

Paquete DataBase.

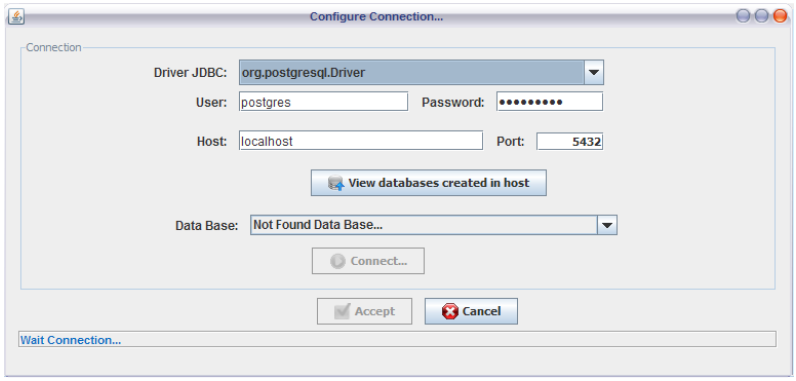
Este paquete es el encargado de permitirle al usuario acceder a la información almacenada en una Base de Datos La conexión se realiza a través de JDBC. Las clases más importantes de este paquete son DBConnectionIcon, que hereda de la clase Icon Esta clase brinda al usuario las operaciones que se pueden realizar en cuanto al manejo de una base de datos ver figura 120,

Figura 120: Operaraciones en la instacia de la clase DBConnectionIcon

Entre las opciones se encuentran *Configure…* la cual instancia la clase connectionWizard, que es la encargada de mostrarle al usuario los parámetros de configuración para establecer la conexión con el SGBD y la base datos ver figura 121, en el se presenta el botón *view database created in host* que establece la conexión de datos postgres y realiza ejecuta un “SELECT * FROM PG_DATABASE WHERE DATISTEMPLATE=FALSE;” que pemite listar las bases de creadas, y por último se escoge en el nombre de a base de datos y con click en el botón *connect* se estable la conexión. Si es posible. Todas las operaciones realizadas en esta ventana se presenta el estado de cada una de ellas en la parte inferior de la ventana, que es una instancia de un JLabel. Ya establecido la

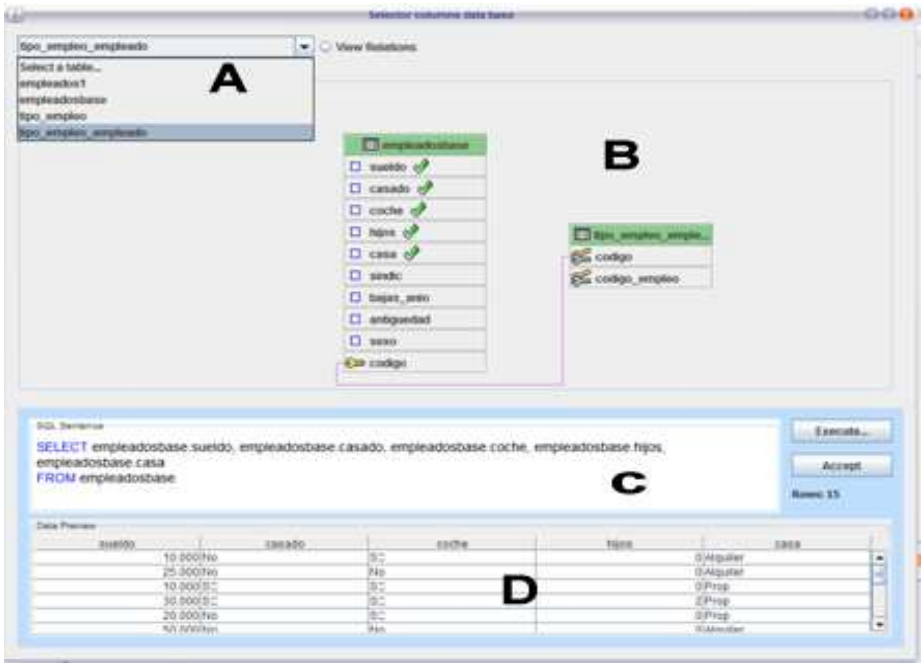
conexión el usuario ejecuta el botón aceptar y se cierra la ventana guardando la conexión establecida para los posteriores pasos.

Figura 121: Instancia de la clase connectionWizard.



Ya establecida la conexión. La instancia de DBConecctionIcon se habilita la operación *Atribute Selecction*, que instancia la clase MySelectorTable. que permite seleccionar las tablas y los campos que se deseen trabajar de una forma grafica. Las tablas encontradas en la base de datos se presentan en JComboBox en el cual el usuario puede dar click sobre la tabla y la tabla se presenta en un área de trabajo ver figura 122 parte **A**.

Figura 122: Instancia de la claseMySelectorTable



El área de trabajo es una instancia de la clase MyJPanelTable, que es la permite mostrar de forma grafica las tablas seleccionadas por el usuario, y se presentan el tipo de relación que hay entre tablas, como la dependencia. Las tablas que son adicionadas al área de trabajo son instancias de la clase Table, clase que como se dijo anteriormente muestra la información de la tabla seleccionada, la información que presenta es el nombre y los atributos, esta clase hereda de un JPanel y los atributos son instancia de la clase Campo que hereda de un JLabel, la clase campo es la encargada de manejar los eventos de selección y desselección de un atributo. Y a medida de ir seleccionando va creando una variable de tipo String que va recolectando los atributos seleccionados ver figura 122 parte **B**.

En la clase MySelectorTable también se encuentra un JEditorPane se muestra una sentencia sql, que se va estableciendo a medida que el usuario va seleccionado los atributos a trabajar (ver figura 122 parte **C)**. Aquí también se puede encontrar un botón que le permite al usuario ver al usuario el resultado de su selección denominado Execute, que llama crea una instancia llamada ScrollableTableModel que permite realizar la consulta con la variable String y devuelve un JTableModel. Este resultado se presenta en forma de tabla ubicado en la parte inferior de la interfaz. (ver figura 122 parte **D)**.

DBConnectionIcon tiene la otra operación denominada *Load* la cual transforma los datos establecidos en la instancia MySelectorTable y son adecuados a una instancia de la clase DataSet, por medio de el método copyData que recibe como parámetro un JTableModel. Y así se estable el conjunto de datos a trabajar. Por último se encuentra la operación *View Data*, muestra una instancia de la clase ViewMetaData que presenta información del conjunto de datos que se ha establecido.

Paquete Clustering

Es este paquete se encuentran todos las clases que manejan la parte grafica de los algoritmos de clustering. Aquí se encuentra la clase ClusteringIcon que hereda de la clase Icon la cual permite al usuario trabajar con los algoritmos implementados y tiene el menú emergente tiene las opciones (ver figura 123):

Configure: es la operación que permite configurar los parámetros necesarios según requiera cada algoritmo.

Run: ejecuta los algoritmos.

Add Attribute Clúster: es la operación encargada de crear en el conjunto de datos un nuevo atributo de tipo categórico denominada cluster en el cual a cada registro se le asigna el valor del clúster al que fue asignado. Creando posteriormente un archivo de formato ARFF con una instancia de la clase FileARFF.

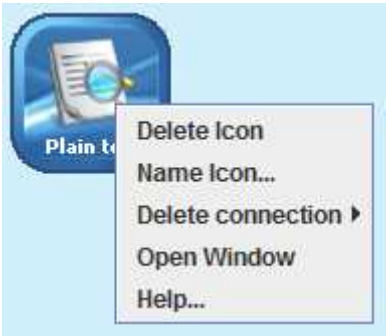
Figura 123: Instancia de la clase ClusteringIcon.

Las operaciones de una instancia ClusteringIcon solo funcionan si hay una línea de conexión entre la instancia y una instancia de FileIcon o DBConecctionIcon, a excepción de las operaciones heredadas por la claseIcon.

Paquete View

Este paquete es el encargado de mostrar los resultados de los algoritmos y ver en algunos graficas del conjunto de datos a trabajar. En este paquete se encuentra la clase ViewIcon que hereda de la clase Icon (ver figura 124). Esta clase permite al usuario tener la operación *Open Window* que como su nombre lo dice abre un ventana que muestra los resultado de la forma en la cual se haya escogido la opción de vista. La opción *Open Window* se habilita solo si el icono tiene una línea de conexión con un ClusteringIcon o un FileIcon o un DBConnection dependiendo de tipo vista escogido. A continuación se presentan los paquetes que se encuentran aquí.

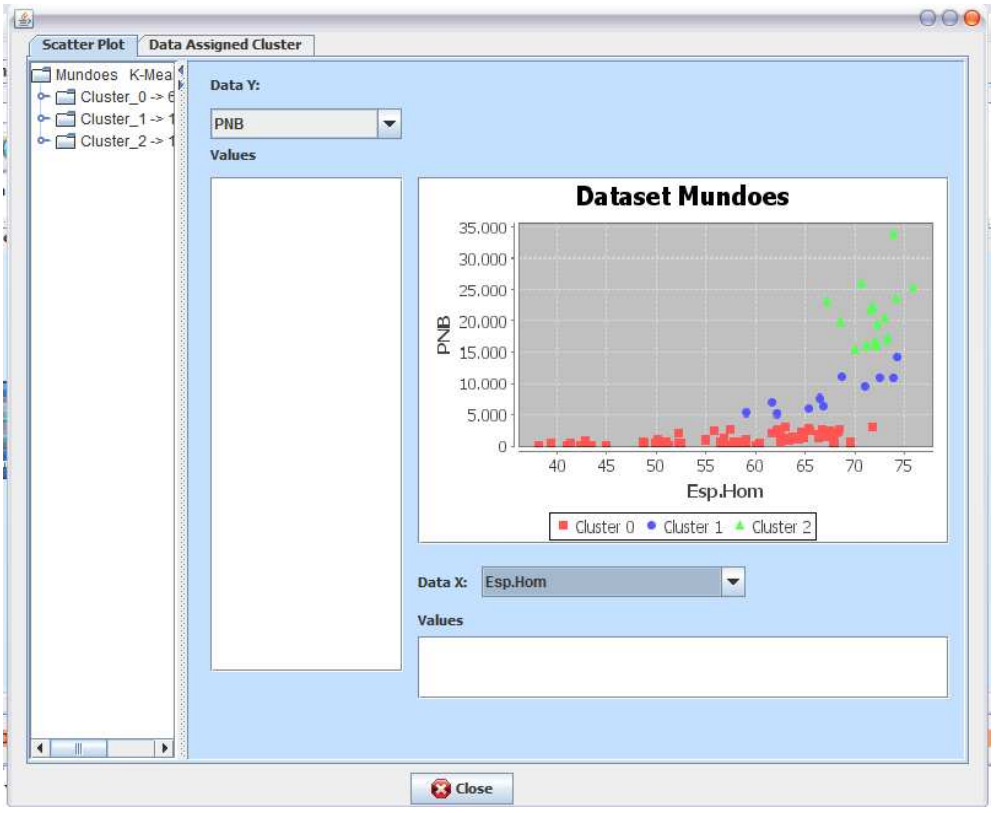
Figura 124: Instancia de la clase ViewIcon.



Paquete ScattePlot: Este paquete es el encargado de contener las clases que permiten ver un diagrama de dispersión con dos ejes. Se presenta una instancia de una clase FrmScatterPlot que hereda de un JFrame e incluye un JTabbedPane,

en la primera pestaña se muestra el grafico ScatterPlot en un PanelScatterPlot que sobrecarga el método paint y dibujando una instancia de la clase Image que es retornada por el método estático ChartFactory.createScatterPlot de la biblioteca JFreeChart y es el encargado de generar la grafica. Esta pestaña también se encuentran dos JComboBox en los cuales el usuario puede elegir el tipo de columna que desea proyectar en la imagen, uno para cada eje. También se encuentra en la parte inferior de cada JComboBox, un JTextArea que presenta información de la columna que esta visualizando y en la parte derecha de la grafica se encuentra un JTree el cual se representa los clúster como ramas del árbol y sus hijos son los valores medios de cada atributo. Ver figura 125. En la segunda pestaña se presenta los registros del conjunto de datos, adicionada la columna *clúster asigment* que indica el numero del clúster asignado. Estas operaciones se pueden realizar solo hay una conexión con instancias de un ClusteringIcon o también se puede presentar los datos de un FileIcon o DBConnectionIcon para mirar la distribución de los datos.

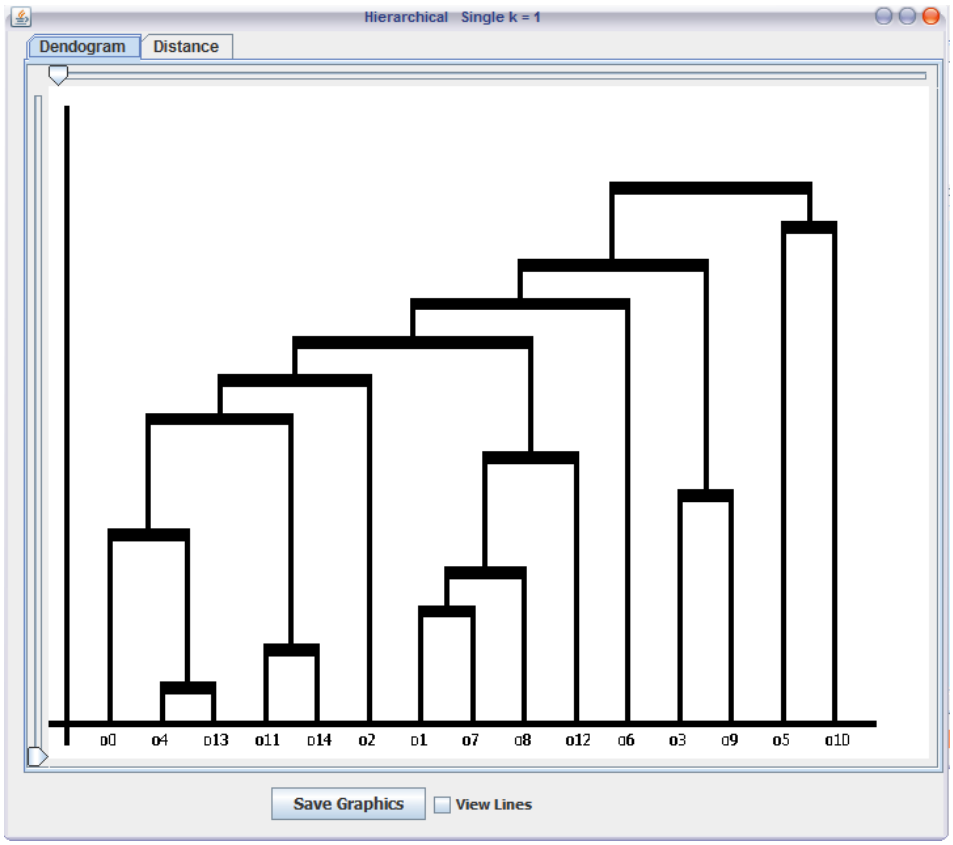
Figura 125: Instancia de la clase FrmViewScatterPlot.



Paquete Dendogram: En este paquete están las clases que permiten ver un dendograma que es el resultado de un algoritmo de clustering jerárquico. Para ello

se presenta una instancia de la clase FrmViewDendogram que es una instancia que hereda de un JFrame e incluye una instancia de la clase PnlGraphicsDendogram que hereda de la clase JPanel y sobrecarga el método paint para que pueda pintar las líneas del dendograma. Los puntos de las líneas del dendograma son asignados en la función obtenerPuntos que recibe una instancia de la clase nodo que es el nodo raíz de un árbol AVL y empieza hacer un recorrido preorden y estable los puntos. Ver figura 126.

Figura 126: Instancia de la clase FrmViewDendogram.



Paquete PlainText: En este paquete se encuentra la clase FrmPlainText que hereda de la clase JFrame. Esta clase tiene incluido una instancia de la clase JTabbedPane con cinco opciones, ver figura 127. La primera nos sirve nos presenta una instancia de la clase JEditorPane, la cual es configurada para interpretar líneas de html con el método setContentType (“text/html”), toda la información de los resultados de los algoritmos de clustering son almacenados en una variable String y es publicada en la Instancia del JEditorPane con el método setText (String), aquí se presentan 3 botones que permiten al usuario crear archivos pdf, rtf, y html con la información de los resultados. Para ello llaman los eventos de la clase ExportFileFormat, clase la cual utiliza la biblioteca iText. La

otra opción nos presenta el conjunto de datos adicionada una columna denominada *Data Assigment* que presenta el conjunto de datos evaluado en un JTable adicionado como columna el número de clúster al cual fue asignado el registro (ver figura 127), las demás opciones funcionan de forma similar, utilizan instancias de la clase JTable.

Figura 127: Instancia de la clase FrmPlainText en la opción Plain Text.

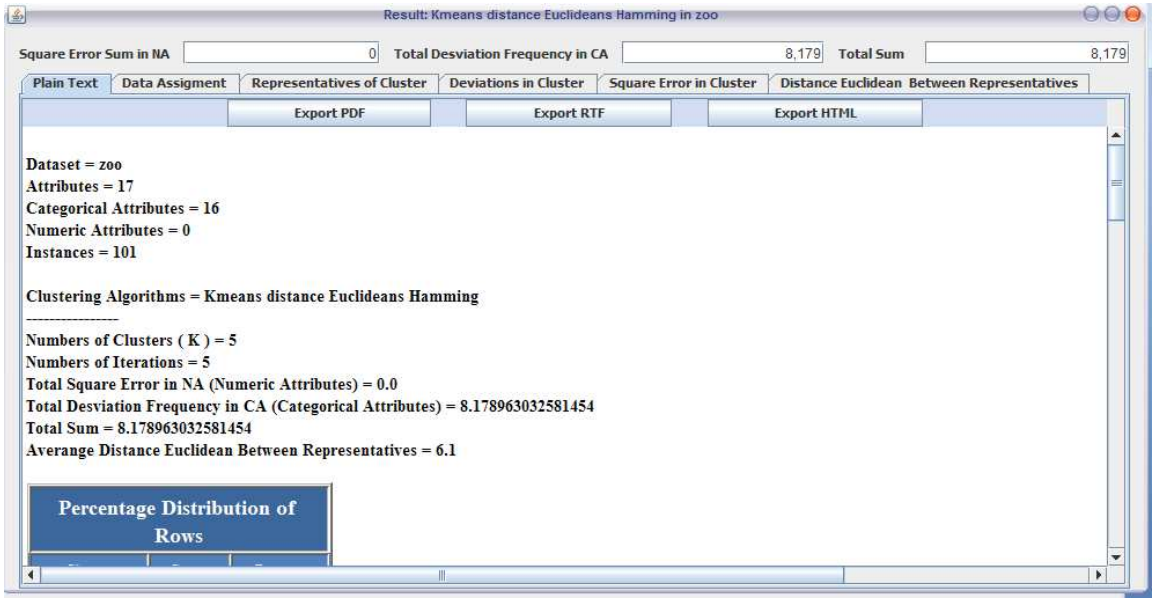
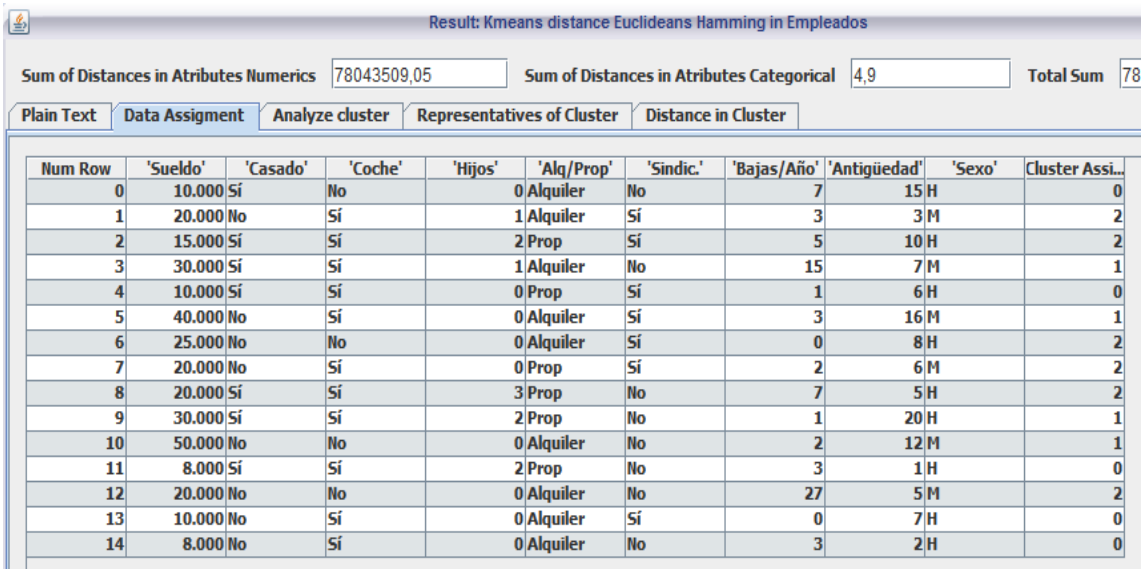


Figura 128: Instancia de la clase FrmPlainText en la opción Data Assigment.



Num Row	'Sueldo'	'Casado'	'Coche'	'Hijos'	'Alq/Prop'	'Sindic.'	'Bajas/Año'	'Antigüedad'	'Sexo'	Cluster Assi...
0	10.000	Sí	No	0	Alquiler	No	7	15	H	0
1	20.000	No	Sí	1	Alquiler	Sí	3	3	M	2
2	15.000	Sí	Sí	2	Prop	Sí	5	10	H	2
3	30.000	Sí	Sí	1	Alquiler	No	15	7	M	1
4	10.000	Sí	Sí	0	Prop	Sí	1	6	H	0
5	40.000	No	Sí	0	Alquiler	Sí	3	16	M	1
6	25.000	No	No	0	Alquiler	Sí	0	8	H	2
7	20.000	No	Sí	0	Prop	Sí	2	6	M	2
8	20.000	Sí	Sí	3	Prop	No	7	5	H	2
9	30.000	Sí	Sí	2	Prop	No	1	20	H	1
10	50.000	No	No	0	Alquiler	No	2	12	M	1
11	8.000	Sí	Sí	2	Prop	No	3	1	H	0
12	20.000	No	No	0	Alquiler	No	27	5	M	2
13	10.000	No	Sí	0	Alquiler	Sí	0	7	H	0
14	8.000	No	Sí	0	Alquiler	No	3	2	H	0

7.2.2. Paquete Algorithms.

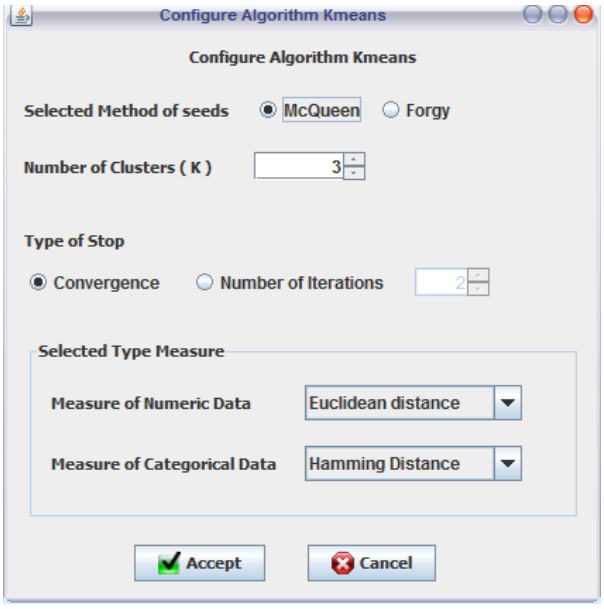
Es este paquete se encuentran todos los algoritmos de clustering implementados en la herramienta y las clases necesarias para su ejecución. A continuación se presenta los paquetes que están contenidos en él.

Paquete Kmeans.

En este paquete se encuentra las clases frmConfigureKmeans y BasicKmeas la cual contiene los métodos que permiten generar clúster a partir de un conjunto de datos.

Clase frmConfigureKmeans: esta clase se encarga de solicitarle al usuario los parámetros necesarios para la ejecución del algoritmo kmean. Esta clase hereda de un JFrame. Utiliza aquí se puede definir el tipo de distancia que se va a utilizar como funciones de distancia entre los elementos. Estas opciones se encuentran dentro de un JComboBox. Al dar click en el botón Accept se asignan los valores establecidos por el usuario a las variables de la instancia de la clase ClusteringIcon (ver fgura 129)

Figura 129: Instancia de la clase frmConfigureKmeans.



Clase BasicKmeans: esta clase contiene métodos que realizan los pasos del algoritmo kmeans. Esta clase hereda de la clase Thread para crear un proceso independiente de la interfaz. El constructor de esta clase recibe como parámetro una instancia de la clase ClusteringIcon, (ver figura 130). Recibe la instancia de la

clase ClusteringIcon porque ahí están los parámetros de configuración definidos por el usuario, además en esa instancia se va a colocar los resultados del algoritmo.

Figura 130: constructor de la clase BasicKmeans

```
public BasicKmeans(ClusteringIcon icon){
        this.icon = icon;
        this.dataset = icon.dataset;
        numIterations = icon.numIte;
    }
```

El algoritmo se empieza a ejecutarse cuando se llama al método run que recibe como parámetro el número de clúster que se desea formar y el número de iteraciones que desea el usuario que se realicen, como se puede ver en la figura 131. Dentro del método run si el numero de iteraciones es igual a cero se lleva el algoritmo no se deja de ejecutar hasta que se llegue a un estado de convergencia, es decir hasta que ningún registro del conjunto de datos cambie de clúster. Si el punto de convergencia es el estimativo de parada, se ejecutara un método kmeans de acuerdo a la combinación de distancias que haya definido el usuario. Además el método run es el encargado de tomar el tiempo de ejecución del algoritmo, tomando el tiempo del sistema con el método estático System.currentTimeMillis que retorna una variable tipo long con el valor del tiempo en milisegundos del sistema. Para saber qué tiempo tardo, se guarda el tiempo inicial, cuando termine el método llamado, se resta con el tiempo final y se muestra en la barra de estado del ContainerWindow. Esta clase contiene los métodos que implementan el algoritmo k-means con diferentes tipos de distancia, el método que se va ejecutar es llamado en la clase run(ver figura 131). Los métodos implementados son:

- ✓ algorithmKmeansEuclideanHamming(kc,this.clusterIcon.forgy,this.clusterIcon.numIte);
- ✓ algorithmKmeansEuclideanProportion(kc,this.clusterIcon.forgy,this.clusterIcon.numIte);
- ✓ algorithmKmeansManhattanHamming(kc,this.clusterIcon.forgy,this.clusterIcon.numIte);
- ✓ algorithmKmeansManhattanProportion(kc,this.clusterIcon.forgy,this.clusterIcon.numIte);
- ✓ algorithmKmeansChebyshevHamming(kc,this.clusterIcon.forgy,this.clusterIcon.numIte);

- ✓ algorithmKmeansChebyshevProportion(kc,this.clusterIcon.forgy,this.clusterIcon.numIte);
- ✓ talgorithmKmeansMahalanobisHamming(kc,this.clusterIcon.forgy,this.clusterIcon.numIte);
- ✓ algorithmKmeansMahalanobisProportion(kc,this.clusterIcon.forgy,this.clusterIcon.numIte);

Figura 131: Metodo run clase BasicKmeans.

```
public void run(int kc,int iteration) {

      String distance="";
      long timeIni = System.currentTimeMillis();

     switch (this.clusterIcon.typeDistance){

            case 0:
                distance = "Euclideans Hamming";
                      this.algorithmKmeansEuclideanHamming(kc,this.clusterIcon.forgy,
                      this.clusterIcon.numIte);
                break;
            case 1:
                distance = "Euclideans Proportion";
                       this.algorithmKmeansEuclideanProportion(kc,this.clusterIcon.forgy,
                       this.clusterIcon.numIte);
                break;
            case 10:
                distance = "Manhattan Hamming";
                      this.algorithmKmeansManhattanHamming(kc,this.clusterIcon.forgy,
                      this.clusterIcon.numIte);
                break;
            case 11:
                distance = "Manhattan Proportion";
                      this.algorithmKmeansManhattanProportion(kc,this.clusterIcon.forgy,
                      this.clusterIcon.numIte);
                break;
             case 20:
                distance="Chebyshev Hamming";
                           this.algorithmKmeansChebyshevHamming(kc,this.clusterIcon.forgy,
                      this.clusterIcon.numIte);
                 break;

continua …
```

```
            case 21:
                distance = "Chebyshev Proportion";
                        this.algorithmKmeansChebyshevProportion(kc,this.clusterIcon.forgy,
                        this.clusterIcon.numIte);
                break;
            case 30:
                 distance = "Mahalanobis Hamming";
                         algorithmKmeansMahalanobisHamming(kc,this.clusterIcon.forgy,
                        this.clusterIcon.numIte);
                break;
             case 31:
                distance = "Mahalanobis Proportion";
                         algorithmKmeansMahalanobisProportion(kc,this.clusterIcon.forgy
                        ,this.clusterIcon.numIte);
                break;
        }

    long timeFin = System.currentTimeMillis() - timeIni;

          ContainerWindow.setStates("Time transcurrido Kmeans distance "+distance+" :
          "+timeFin+" milisegundos" );

     this.clusterIcon.meansCluster =   this.meansCluster;
     this.clusterIcon.numElementsCluster = this.numElementsCluster;
     this.clusterIcon.valuesAttributesCluster = this.valuesAttributesCluster;
    // se calcula la desviacion estandar
     dataset.calculateDesviationStandand(this.clusterIcon);
     clusterIcon.distanceElemtsCluster = this.distanceElemtsCluster;
     this.clusterIcon.nameAlgorithm = "Kmeans distance "+distance;
     clusterIcon.nomImagen = "/image/Algorithms/kmeansIcon.png";
     clusterIcon.repaint();

  }
```

Todas las diferentes combinaciones de distancia separadas en métodos se hizo para mejorar el tiempo de ejecución, los pasos básicos se conservan la única diferencia está en la llamada de los métodos de distancia (ver figura 132 en *). Además se tiene una matriz de objetos (Object[ ][ ]), meansClusterAux en donde se van acumulando los valores de los objetos asignados según el numero de clúster con el fin de que al final al calcular los nuevos centroides se divida este dichos valores acumulados por el numero de objetos asignados en al clúster.

Figura 132: Método algorithmKmeansEuclideanHamming de la clase BasicKmeans.

```
public void algorithmKmeansEuclideanHamming(int kc,boolean random,int iteration){
    int antAssigCluster[] = null;
    int i,j;
    boolean converge;

     // selecciona las semillas segun k: datos,num k, true = aleatorio ó false = los primeros
     assigCluster = new int[dataset.numInstances];
     valuesAttributesCluster = new Object [kc][];
     meansCluster =  this.selectSeeds(dataset,kc,random);
     int numIteration = 0;

     double  distance;
     double  menorDistancia;
     converge = false;

     while(!converge &&  iteration != numIteration){
        converge = true;
        antAssigCluster = assigCluster;
        assigCluster = new int[dataset.numInstances];
        numElementsCluster = new int[kc];
        meansClusterAux = new Object[kc][dataset.numAtributos];
        distanceElemtsCluster = new double[kc];

       //calcular la distancia de todos los objeto con los centros de cluster
        for(i=0;i < dataset.numInstances;i++ ){

        //* calcular distancia del obj[i] con el primer cluster y se guanrda como el menor
          menorDistancia=Distance.measureDistanceEuclideanHamming(dataset.instancias[i],
         meansCluster[antAssigCluster[i]],dataset.columnsNumeric,dataset.ini);
          assigCluster[i] = antAssigCluster[i];

          // * calculo las distancias del resto de Clusters
          for(j=0;j<kc;j++){
                    distance=Distance.measureDistanceEuclideanHamming(dataset.instancias[i],
                    meansCluster[j],dataset.columnsNumeric,dataset.ini);
                   // reviso la menor Distancia
                  if(distance < menorDistancia){
                     menorDistancia=distance;
                    assigCluster[i]=j;
                   );
              converge = false;
            }
          }

          Continua…
```

```
            Object [] auxInstance;
           //Actulaizando Vectores
             auxInstance = dataset.instancias[i];
           //inicializo la matriz de Means de c/clustel

          for(j=dataset.ini;j<dataset.numAtributos ;j++){
              if( (Boolean) dataset.columnsNumeric[j] ){
                if( meansClusterAux[assigCluster[i]][j] == null){
                   //meansCluster[assigCluster[i]][j] = (Double) auxInstance[j];
                      meansClusterAux[assigCluster[i]][j]=
                    Double.parseDouble(auxInstance[j].toString());
                }else{
                      meansClusterAux[assigCluster[i]][j]=((Double)
                    meansClusterAux[assigCluster[i]][j])
                    +Double.parseDouble(auxInstance[j].toString());
                }
              }else{

                if( meansClusterAux[assigCluster[i]][j] == null){
                   //se crea un array del tam de los posibles valores del atributo y
                   // se lo asigna en el espacio dl atributo del centro de grupo
                    int [] aux  =  new int[((Vector) dataset.valueCategorical[j]).size()];

                   aux[((Vector) dataset.valueCategorical[j]).indexOf(auxInstance[j])]=1;
                   meansClusterAux[ assigCluster[i] ][j] = aux;
                }else{
                    ((int[])meansClusterAux[assigCluster[i]][j])[((Vector)
                    dataset.valueCategorical[j]).indexOf(auxInstance[j] )]++;
                }
              }
           }
           numElementsCluster[assigCluster[i]]++;
        }// fin for

     this.clusterIcon.totalIterations =numIteration;
     this.clusterIcon.assigCluster = assigCluster;
  }
```

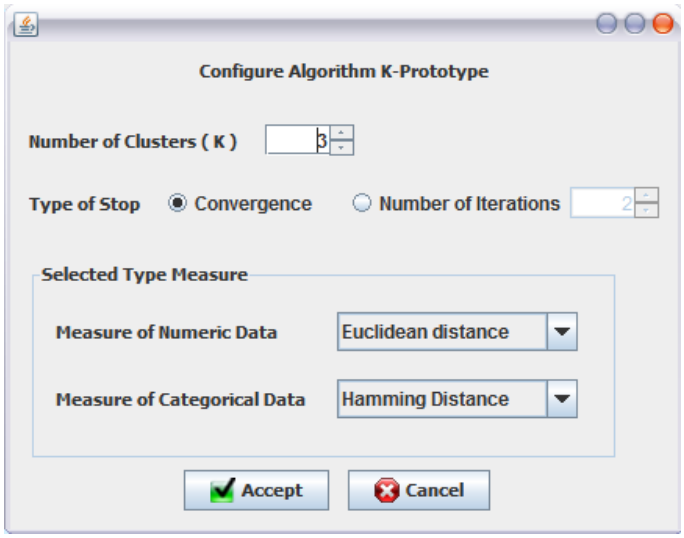
Paquete kprototype.

En este paquete se encuentra las clases frmConfigureKprototype y KPrototype la cual contiene los métodos que permiten generar clúster a partir de un conjunto de datos.

Clase frmConfigureKprototype: esta clase se encarga de solicitarle al usuario los parámetros necesarios para la ejecución del algoritmo kprototype. Esta clase hereda de un JFrame. Utiliza aquí se puede definir el tipo de distancia que se va a utilizar como funciones de distancia entre los elementos. Estas opciones se encuentran dentro de un JComboBox. Al dar click en el botón Accept se asignan

los valores establecidos por el usuario a las variables de la instancia de la clase ClusteringIcon (ver figura 133).

Figura 133: Instancia de la clase frmConfigureKprototype.



Clase KPrototype: esta clase contiene métodos que realizan los pasos del algoritmo KPrototype. Esta clase hereda de la clase Thread para crear un proceso independiente de la interfaz. El constructor de esta clase recibe como parámetro una instancia de la clase ClusteringIcon, (ver figura 134). Recibe la instancia de la clase ClusteringIcon porque ahí están los parámetros de configuración definidos por el usuario, además en esa instancia se va a colocar los resultados del algoritmo.

Figura 134: constructor clase KPrototype.

```
public KPrototype(ClusteringIcon icon){
      this.icon = icon;
       this.dataset = icon.dataset;
 }
```

El algoritmo se empieza a ejecutarse cuando se llama al método run que recibe como parámetro el número de clúster que se desea formar y el numero de iteraciones que desea el usuario que se realicen, como se puede ver en la figura 135. Dentro del método run si el numero de iteraciones es igual a cero se lleva el algoritmo no se deja de ejecutar hasta que se llegue a un estado de convergencia, es decir hasta que ningún registro del conjunto de datos cambie de clúster. Si el punto de convergencia es el estimativo de parada, se ejecutara un método kprototype de acuerdo a la combinación de distancias que haya definido el

usuario. Además el método run es el encargado de tomar el tiempo de ejecución del algoritmo, tomando el tiempo del sistema con el método estático System.currentTimeMillis que retorna una variable tipo long con el valor del tiempo en milisegundos del sistema. Para saber qué tiempo tardo, se guarda el tiempo inicial, cuando termine el método llamado, se resta con el tiempo final y se muestra en la barra de estado del ContainerWindow. Esta clase contiene los métodos que implementan el algoritmo k-prototype con diferentes tipos de distancia, el método que se va ejecutar es llamado en la clase run(ver figura 135). Los métodos implementados son:

- ✓ algorithmKprototypeEuclideanHamming(kc, this.clusterIcon.numIte);
- ✓ algorithmKprototypeEuclideanProportion(kc, this.clusterIcon.numIte);
- ✓ algorithmKprototypeManhattanHamming(kc, this.clusterIcon.numIte);
- ✓ algorithmKprototypeManhattanProportion(kc, this.clusterIcon.numIte);
- ✓ algorithmKprototypeChebyshevHamming(kc, this.clusterIcon.numIte);
- ✓ algorithmKprototypeChebyshevProportion(kc, this.clusterIcon.numIte);
- ✓ talgorithmKprototypeMahalanobisHamming(kc, this.clusterIcon.numIte);
- ✓ algorithmKprototypeMahalanobisProportion(kc, this.clusterIcon.numIte);

Figura 135: función run de la clase KPrototype.

```
public void run(int kc,int iteration) {

    long timeIni = System.currentTimeMillis();
    String distance="";

   switch (this.clusterIcon.typeDistance){
          case 0:
               distance = "KPrototype Hamming";
               this.algorithmKprototypeEuclideanHamming(kc,this.clusterIcon.numIte);
              break;
          case 1:
               distance = "Euclideans Proportion";
               this.algorithmKprototypeEuclideanProportion(kc,this.clusterIcon.numIte);
              break;

   continua…..
```

```
            case 10:
                distance = "Manhattan Hamming";
                this.algorithmKprototypeManhattanHamming(kc,this.clusterIcon.numIte);
               break;
           case 11:
                distance = "Manhattan Proportion";
                this.algorithmKprototypeManhattanProportion(kc,this.clusterIcon.numIte);
               break;
            case 20:
                distance = "Chebyshev Hamming";
                this.algorithmKprototypeChebyshevHamming(kc,this.clusterIcon.numIte);
               break;
           case 21:
                distance = "Chebyshev Proportion";
                this.algorithmKprototypeChebyshevProportion(kc,this.clusterIcon.numIte);
               break;
           case 30:
                 distance = "Mahalanobis Hamming";
                 this.algorithmKprototypeMahalanobisHamming(kc,this.clusterIcon.numIte);
               break;
            case 31:
                 distance = "Mahalanobis Proportion";
                 this.algorithmKprototypeMahalanobisProportion(kc,this.clusterIcon.numIte);
               break;
       }
  long timeFin =System.currentTimeMillis() - timeIni;

  ContainerWindow.setStates("Time transcurrido Kmeans  distance "+distance+" : "+timeFin+"
  milisegundos" );
    this.clusterIcon.meansCluster =   this.meansCluster;
    this.clusterIcon.numElementsCluster = this.numElementsCluster;
    this.clusterIcon.valuesAttributesCluster=this.valuesAttributesCluster;
    dataset.calculateDesviationStandand(clusterIcon);
    this.clusterIcon.nameAlgorithm = "KPrototype distance "+distance;
    clusterIcon.nomImagen = clusterIcon.nameFigure;
    clusterIcon.repaint();
}
```

Al igual como se hizo para el algoritmo k-means se mantienen los pasos básicos en los métodos la única diferencia está en la llamada de los métodos de distancia (ver figura 136 en *).

Figura 136: Método algorithmKprototypeEuclideanHamming de la clase Kprototype.

```
public void algorithmKprototypeEuclideanHamming(int kClusters,int numIteration) {
      int i,j,k;
       double minDistance;
       double distance;
       int formerCluster,newCluster;
       meansCluster = new Object[kClusters][dataset.numAtributos];
       valuesAttributesCluster = new Object[kClusters][dataset.numAtributos];

      //  inicializar variables
       for (i=0; i<kClusters;i++) {
          for (j = dataset.ini; j < dataset.numAtributos; j++) {
             if( (Boolean) dataset.columnsNumeric[j] ){
                Double  aux = new Double(0);
                valuesAttributesCluster[i][j] = aux;
             }else{
                 valuesAttributesCluster[i][j] = new int [ ((Vector) dataset.valueCategorical[j]).size()] ;
             }
          }
       }

      //1. Iniciar  k-partición de  D en forma aleatoria.
       numElementsCluster = new int[kClusters];
       assignmentCluster = this.ramdomAssignmentCluster(kClusters,dataset);
       boolean converge = false;

       while(!converge && iteration!=numIteration){
          converge = true;
           //calcular la distancia de todos los objeto con los centros de cluster
          for(i=0;i < dataset.numInstances; i++ ){

             //calcular distancia del obj[i] con el primer cluster y se guanrda como el menor
             minDistance = Distance.measureDistanceEuclideanHamming(dataset.instancias[i],
                         meansCluster[assignmentCluster[i]],dataset.columnsNumeric,dataset.ini);
             newCluster  = assignmentCluster[i];
            );

             // calculo las distancias del resto de Clusters
             for(k=0;k<kClusters;k++){
                distance = Distance.measureDistanceEuclideanHamming(dataset.instancias[i],
                        meansCluster[k],dataset.columnsNumeric,dataset.ini);
             // reviso la menor Distancia
                if(distance< minDistance){
                   minDistance = distance;
                   newCluster=k;
                   //System.out.println("estaba en "+assignmentCluster[i] +" cambio "+k);
                }
             }

   Continua…
```

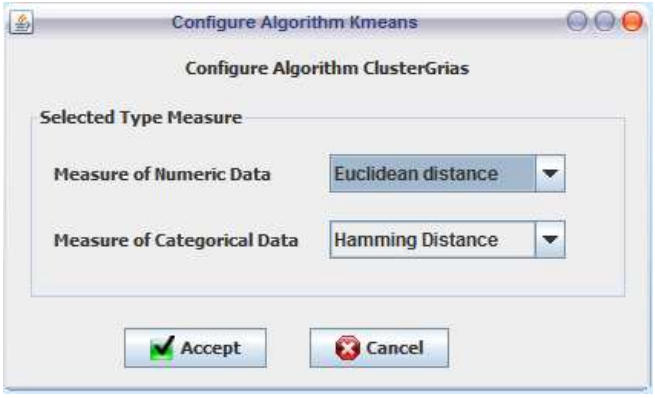
```
            // miro si el objeto cambio de cluster
             if( newCluster != assignmentCluster[i] ){
               // reorganizo el grupo y encuentro los nuevos centro de grupo
                formerCluster = assignmentCluster[i];
                ajustarValoresClusterRemove(formerCluster,dataset.instancias[i]);
                assignmentCluster[i] = newCluster;
                ajustarValoresClusterAdd(newCluster,dataset.instancias[i]);
                converge = false;
             }
           }//end for
           iteration++;
       }// end while
   }
```

Paquete wayCluster.

En este paquete se encuentra las clases frmConfigureWayCLuster y WayCluster la cual contiene los métodos que permiten generar clúster a partir de un conjunto de datos.

Clase frmConfigureWayCluster: esta clase se encarga de solicitarle al usuario los parámetros necesarios para la ejecución del algoritmo wayCluster (Ver figura 137). Esta clase hereda de un JFrame. Utiliza aquí se puede definir el tipo de distancia que se va a utilizar como funciones de distancia entre los elementos. Estas opciones se encuentran dentro de un JComboBox. Al dar click en el botón Accept se asignan los valores establecidos por el usuario a las variables de la instancia de la clase ClusteringIcon

Figura 137: Instancia de la clase frmConfigureWayCluster.



Clase WayCluster: esta clase contiene métodos que realizan los pasos del algoritmo KPrototype. Esta clase hereda de la clase Thread para crear un proceso independiente de la interfaz. El constructor de esta clase recibe como parámetro una instancia de la clase ClusteringIcon,(ver figura 138). Recibe la instancia de la clase ClusteringIcon porque ahí están los parámetros de configuración definidos por el usuario, además en esa instancia se va a colocar los resultados del algoritmo.

Figura 138: constructor clase WayCluster.

```
public wayCluster(ClusteringIcon clusterIcon){

    this.clusterIcon = clusterIcon;
    dataset = clusterIcon.dataset;

}
```

El algoritmo se empieza a ejecutarse cuando se llama al método run (ver en la figura 121). se ejecutara un método estático de la clase distancia el cual nos devuelve la matriz de distancia, de acuerdo a la combinación de distancias que haya definido el usuario. Además el método run es el encargado de tomar el tiempo de ejecución del algoritmo, tomando el tiempo del sistema con el método estático System.currentTimeMillis que retorna una variable tipo long con el valor del tiempo en milisegundos del sistema. Para saber qué tiempo tardo, se guarda el tiempo inicial, cuando termine el método llamado, se resta con el tiempo final y se muestra en la barra de estado del ContainerWindow. La matriz de distancia que se necesita se la obtiene de los métodos estáticos de la clase Distancia que retornan una matriz Double [ ] [ ](ver figura 139), los cuales se listan a continuación:

- ✓ Distance.calculateDistanceMatrixEuclideanHamming(clusterIcon.dataset);
- ✓ Distance.calculateDistanceMatrixEuclideanProportion(clusterIcon.dataset);
- ✓ Distance.calculateDistanceMatrixMahattanHamming(clusterIcon.dataset);
- ✓ Distance.calculateDistanceMatrixManhattanProportion(clusterIcon.dataset);
- ✓ Distance.calculateDistanceMatrixChebyshevHamming(clusterIcon.dataset);
- ✓ Distance.calculateDistanceMatrixChebyshevProportion(clusterIcon.dataset);
- ✓ Distance.calculateDistanceMatrixMahalanobisHamming(clusterIcon.dataset)
- ✓ Distance.calculateDistanceMatrixMahalanobisProportion(clusterIcon.dataset);

Figura 139: función run de la clase WayCluster.

```
public void run() {
      long timeIni = System.currentTimeMillis();
      Double[][] matrix = null;

      switch (clusterIcon.typeDistance){
         case 00:
            matrix = Distance.calculateDistanceMatrixEuclideanHamming(clusterIcon.dataset);
            break;
        case 01:
            matrix = Distance.calculateDistanceMatrixEuclideanProportion(clusterIcon.dataset);
            break;
        case 10:
            matrix = Distance.calculateDistanceMatrixMahattanHamming(clusterIcon.dataset);
            break;
        case 11:
            matrix = Distance.calculateDistanceMatrixManhattanProportion(clusterIcon.dataset);
            break;
        case 20:
            matrix = Distance.calculateDistanceMatrixChebyshevHamming(clusterIcon.dataset);
            break;
        case 21:
            matrix = Distance.calculateDistanceMatrixChebyshevProportion(clusterIcon.dataset);
            break;
        case 30:
            matrix = Distance.calculateDistanceMatrixMahalanobisHamming(clusterIcon.dataset);
            break;
        case 31:
            matrix = Distance.calculateDistanceMatrixMahalanobisProportion(clusterIcon.dataset);
            break;
      }
      this.algorithmWayCluster(matrix);
       long timeFin =System.currentTimeMillis() - timeIni;
        ContainerWindow.setStates("Time transcurrido ClusterNew: "+timeFin+" milisegundos" );
      this.clusterIcon.meansCluster =   this.meansCluster;
      this.clusterIcon.numElementsCluster = this.numElementsCluster;
      this.clusterIcon.valuesAttributesCluster = this.valuesAttributesCluster;
      dataset.calculateDesviationStandand(clusterIcon);
      this.clusterIcon.nomImagen = this.clusterIcon.nameFigure;
      this.clusterIcon.repaint();

   }
```

A diferencia de los algoritmos anteriores, el algoritmo WayClase solo tiene un método que tiene los pasos y es el método algorithmClusterWay (ver figura 140), el cual recibe como parámetros una matriz de de Double[ ][ ], que son las distancias entre registros.

Figura 140: Método algorithmWayCluster de la clase WayCluster.

```
private void algorithmWayCluster (Double[][] matrix ){
        Vector activos;
        int i,j,k;
        activos = new Vector();
        double distance;
        boolean [] objectActive = new boolean[dataset.numInstances];
        Vector clustersIndices = new Vector();

        assigmentCluster = new int[dataset.numInstances];
        double [] menorAdd = new double[dataset.numInstances];

        double menor;

        for(i=0; i<dataset.numInstances;i++  )
           menorAdd[i]= Double.MAX_VALUE;

        k=0;

        double auxDistance;
        for(i =0; i < dataset.numInstances;i++){
              menor = Double.MAX_VALUE;

           // recorer la columna de distancia del objeto y encontrar la menor distancia
           for(j=0;j<dataset.numInstances;j++  ){

              if ( i!=j) {
                 if (matrix[i][j] != null)
                    auxDistance = matrix[i][j];
                 else
                    auxDistance = matrix[j][i];

                 if (menor > auxDistance)
                    menor = auxDistance;
              }
           }


      // verificar si el objeto fue asignado
           if( menorAdd[i] == Double.MAX_VALUE ){
              assigmentCluster[i] = i;
              menorAdd[i] = 0;
              // si no fue asignado crear nuevo cluster
              clustersIndices.add(i);
              k++;
              //System.out.println(" creo nuevo cluster");
           }

Continua….
```

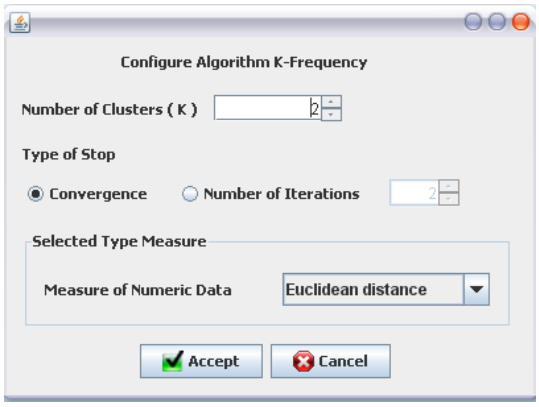
```
    for(j=0; j< dataset.numInstances;j++ ){

            if (i!=j) {
               if (matrix[j][i] != null)
                  auxDistance = matrix[j][i];
               else
                  auxDistance = matrix[i][j];


               // si la distancia del objeto j es igual a menor y
               // la distanca a la cual fue asignada es menor

               if (menor == auxDistance && menor < menorAdd[j]) {
                  assigmentCluster[j] = assigmentCluster[i];
                  menorAdd[j] = menor;
               }
            }

         }

      }

      // calcular centroides

      this.clusterIcon.k = k;
      this.calculateNewsCentoids(clustersIndices);
      this.clusterIcon.assigCluster = assigmentCluster;
   }
```

Paquete KFrequency.

En este paquete se encuentra las clases frmConfigureKFrequency y KFrequency la cual contiene los métodos que permiten generar clúster a partir de un conjunto de datos.

Clase frmConfigureKFrequency: esta clase se encarga de solicitarle al usuario los parámetros necesarios para la ejecución del algoritmo KFrequency. Esta clase hereda de un JFrame. Utiliza aquí se puede definir el tipo de distancia que se va a utilizar como funciones de distancia entre los elementos, aquí solo se definirá la distancia entre atributos numéricos (ver figura 141) ya que este algoritmo su esencia está en una forma particular de obtener la distancia entre atributos categóricos . Estas opciones se encuentran dentro de un JComboBox. Al dar click en el botón Accept se asignan los valores establecidos por el usuario a las variables de la instancia de la clase ClusteringIcon

Figura 141: Instancia de la clase frmConfigureKFrequency.



Clase KFrequency: esta clase contiene métodos que realizan los pasos del algoritmo KFrequency. Esta clase hereda de la clase Thread para crear un proceso independiente de la interfaz. El constructor de esta clase recibe como parámetro una instancia de la clase ClusteringIcon, (ver figura 142). Recibe la instancia de la clase ClusteringIcon porque ahí están los parámetros de configuración definidos por el usuario, además en esa instancia se va a colocar los resultados del algoritmo.

Figura 142: constructor clase KFrequency.

```
public KFrequency(ClusteringIcon icon){

    this.clusterIcon = icon;
    this.dataset = icon.dataset;
}
```

El algoritmo se empieza a ejecutarse cuando se llama al método run que recibe como parámetro el número de clúster que se desea formar y el número de iteraciones que desea el usuario que se realicen, como se puede ver en la figura 143. Dentro del método run si el numero de iteraciones es igual a cero se lleva el algoritmo no se deja de ejecutar hasta que se llegue a un estado de convergencia, es decir hasta que ningún registro del conjunto de datos cambie de clúster. Si el punto de convergencia es el estimativo de parada, se ejecutara un método kfequency de acuerdo al tipo de distancia numérica que haya definido el usuario. Además el método run es el encargado de tomar el tiempo de ejecución del algoritmo, tomando el tiempo del sistema con el método estático System.currentTimeMillis que retorna una variable tipo long con el valor del tiempo en milisegundos del sistema. Para saber qué tiempo tardo, se guarda el tiempo inicial, cuando termine el método llamado, se resta con el tiempo final y se muestra en la barra de estado del ContainerWindow. Esta clase contiene los

métodos que implementan el algoritmo kfrequency con diferentes tipos de distancia numerica, el método que se va ejecutar es llamado en la clase run(ver figura 143). Los métodos implementados son:

- ✓ algorithmKFrequencyEuclidean (kc, this.clusterIcon.numIte);
- ✓ algorithmKFrequencyManhattan (kc, this.clusterIcon.numIte);
- ✓ algorithmKFrequencyChebyshev (kc, this.clusterIcon.numIte);
- ✓ talgorithmKFrequencyMahalanobis (kc, this.clusterIcon.numIte);

Figura 143: función run de la clase KFrequency.

```
public void run(int kc,int iteration) {

       String distance="";
       long timeIni = System.currentTimeMillis();

      switch (clusterIcon.typeDistance){
         case 0:
            this.algorithmKFrequencyEuclidean(kc,clusterIcon.numIte);
            distance = "Euclideans Frequency";
            break;
         case 1:
            distance = "Manhattan Frequency";
            this.algorithmKFrequencyManhattan(kc,clusterIcon.numIte);
            break;
         case 2:
            distance = "Chebyshev Frequency";
            this.algorithmKFrequencyChebyshev(kc,clusterIcon.numIte);
            break;
         case 3:
            distance = "Mahalanobis Frequency";
            this.algorithmKFrequencyMahalanobis(kc,clusterIcon.numIte);
            break;
      }

     long timeFin =System.currentTimeMillis() - timeIni;
     ContainerWindow.setStates("Time transcurrido KHistogram: "+timeFin+" milisegundos" );

       this.clusterIcon.meansCluster =   this.meansCluster;
       this.clusterIcon.numElementsCluster = this.numElementsCluster;
       this.clusterIcon.valuesAttributesCluster = this.valuesAttributesCluster;
       dataset.calculateDesviationStandand(clusterIcon);
       this.clusterIcon.nameAlgorithm = "KHistogram distance "+distance;
       clusterIcon.nomImagen = "/image/Algorithms/kfrequencyIcon.png";
       clusterIcon.repaint();
   }
```

Al igual como se hizo para el algoritmo k-means se mantienen los pasos básicos en los métodos la única diferencia está en la llamada de los métodos de distancia (ver figura 144 en *).

Figura 144: Método algorithmKFrequencyEuclidean de la clase KFrequency.

```
public void algorithmKFrequencyEuclidean(int kClusters,int iteration){
      int i,j,k;
       double minDistance;
       double distance;
       int formerCluster,newCluster;

       meansCluster = new Object[kClusters][dataset.numAtributos];
       valuesAttributesCluster = new Object[kClusters][dataset.numAtributos];

       for(i=0; i<kClusters;i++ ){

          for(j=dataset.ini ; j< dataset.numAtributos;j++  ){

             if((Boolean) dataset.columnsNumeric[j] ){
               valuesAttributesCluster[i] [j] = new Double("0");
             }else{
                       valuesAttributesCluster[i] [j] =
                       new int[ ( (Vector)this.dataset.valueCategorical[j]).size()];
             }
          }
       }
       // 1. tomar los k primeros objetos como semilla

       for (i=0; i<kClusters;i++) {
          meansCluster[i] = dataset.instancias[i].clone();
       }

      //2. calcular la distancia de las semillas las instancias y asignar la semillas de menor valor.
       numElementsCluster = new int[kClusters];
       assignmentCluster = new int[dataset.numInstances];

       assignmentClusterMoreSimilar(kClusters, clusterIcon.typeDistance);

       boolean converge = false;

       // 3. Mientras los objetos cambien de cluster
       while(!converge && iteration != iterations){
          converge = true;
            //4. calcular la distancia de todos los objeto con los  clusters
          for(i=0;i < dataset.numInstances;i++ ){

   continua…
```

```
          //5. calcular distancia del obj[i] con el cluster anteriormente asignado y
          //   se guarda como el menor

          *  minDistance = calculateDistanceEuclidean(dataset.instancias[i],
           assignmentCluster[i],dataset.columnsNumeric,numElementsCluster[assignmentCluster[i]]);
            newCluster  = assignmentCluster[i];

            //6. calculo las distancias del resto de Clusters
            for(k=0;k<kClusters;k++){
              * distance = calculateDistanceEuclidean(dataset.instancias[i],
                        k,dataset.columnsNumeric,numElementsCluster[k]);

              // revisar la menor Distancia
               if(distance< minDistance){
                  minDistance = distance;
                  newCluster=k;
               }
            }

            //7. mirar si el objeto cambio de cluster
            if( newCluster != assignmentCluster[i] ){
              //8. reorganizar el grupo y encuentro los nuevos centro de grupo
               formerCluster = assignmentCluster[i];

               ajustarValoresClusterRemove(formerCluster,dataset.instancias[i]);
               assignmentCluster[i] = newCluster;
               ajustarValoresClusterAdd(newCluster,dataset.instancias[i]);
               converge = false;
            }

          }//end for

           iterations++;

       }// end while

   }
```

El método para calcular distancia para el caso de algorithmKFrequencyEuclidean se denomina de calculateDistanceEuclidean, que se encarga de medir la distancia de los atributos categóricos utilizando la frequency del valor del atributo del objeto a evaluar que estos tengan dentro de un cluster. Esto se logra gracias a la matriz de tipo Object [ ] [ ] denominada valuesAttributesCluster, que guarda para los atributos numéricos un acumulado de los valores encontrados y en las columnas de los atributos categóricos tiene un vector de tipo int (int []), en el cual almacena el conteo de los valores encontrados en ese atributos (ver figura 145). Los demás métodos utilizados para medir distancia tienen los mismos pasos, lo único que varia es la fórmula utilizada para el cálculo distancia en los atributos numéricos.

Figura 145: Método calculateDistanceEuclidean de la clase KFrequency.

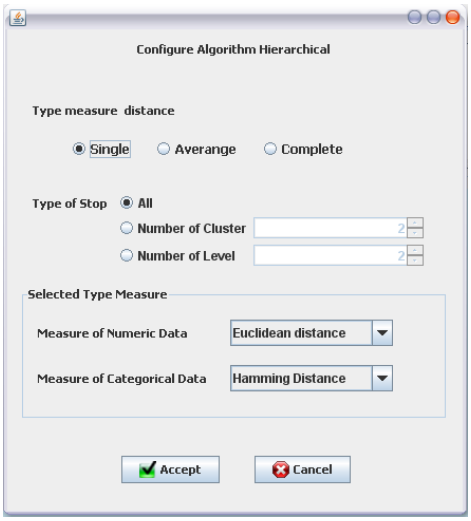
```
private double calculateDistanceEuclidean(Object[] obj1, int cluster, Object [] columnNumeric, int
numElemntsCluster) {

      double distance = 0;
      double valorInterno = 0;
      double valorNominal = 0;
      int  auxModa[];

      // sumo todos los valores   (x1 - x2)elevado al 2 + (y1 - y2) elevado al 2 .....
       for(int i=dataset.ini; i< obj1.length ; i++ ){
         if((Boolean) columnNumeric[i] )
            valorInterno += Math.pow( (Double) obj1[i]- (Double) obj2[i] ,2);
         else{
            auxModa = (int[]) this.valuesAttributesCluster[cluster][i];
           valorNominal+=(1 - (auxModa[   ((Vector)dataset.valueCategorical[i] ).indexOf(obj1[i] )]
          / (double)numElemntsCluster  ));
         }
       }
      // aplico la raiz cuadrada
      distance = (Math.sqrt(valorInterno) + valorNominal);
      return distance;
   }
```

Paquete Hierarchical

En este paquete se encuentran las clases HierachicalBasic, FrmConfigureHierarchical y Nodo la primera clase contiene los métodos que permiten generar un árbol o dendograma a partir de un conjunto de datos haciendo uso de la clase Nodo. La segunda es la que permite configurar al usuario los parámetros de los algoritmos.

Clase FrmConfigureHierarchical: Permite al usuario establecer el tipo de algoritmo jerárquico a utilizar, presentando estas opciones con JRadioButton, también le permite al usuario definir el tipo de parada del algoritmo, también le permite escoger el tipo función de distancia a utilizar para los datos numéricos y también para el tipo de distancia entre atributos categóricos (ver figura 146). Presionando el botón aceptar, se establecen dichos parametros definidos, dentro de la instancia de la clase ClusteringIcon

Figura 146: Instancia de la clase FrmConfigureHierachical.



Clase HierarchicalBasic: Esta clase contiene métodos que realizan los pasos básicos de la técnica de clustering jerárquico aglomerativo. Esta clase hereda de la clase Thread para crear un proceso independiente de la interfaz. El constructor de esta clase recibe como parámetro una instancia de la clase ClusteringIcon, (ver figura 147). Recibe la instancia de la clase ClusteringIcon porque ahí están los parámetros de configuración definidos por el usuario, además en esa instancia se va a colocar los resultados del algoritmo.

Figura 147: constructor de la clase HierarchicalBasic.

```
public HierarchicalBasic(ClusteringIcon clusterIcon){
        this.clusterIcon = clusterIcon;
       dataset = clusterIcon.dataset;
    }
```

El algoritmo inicia invocando al método run que no requiere parámetros, evalúa la variable typeMeasure de la instancia ClusteringIcon con una operación switch que determina la forma de hacer la medición entre grupos. Según este valor se ejecuta su respectivo algoritmo. Es decir si el valor es igual a 1 se ejecuta el algoritmo que hace la medición entre grupos single o minina, si el valor es de 2 se ejecuta el algoritmo que hace la medición entre grupos mediante el promedio de distancias y si el valor es igual a 3 se ejecuta el algoritmo que hace la medición entre grupos denominada máxima o completa. Al finalizar los pasos es asignado en la variable nodoRaiz de la instancia de ClusteringIcon. Esta variable hereda de la clase nodo y asigna el nodo raíz resultado del método ejecutado que es el que permite generar el dendograma. Además el método run es el encargado de tomar el

tiempo de ejecución del algoritmo, tomando el tiempo del sistema con el método estático System.currentTimeMillis que retorna una variable tipo long con el valor del tiempo en milisegundos del sistema. Para saber qué tiempo tardo, se guarda el tiempo inicial, cuando termine el método llamado se resta con el tiempo final y se muestra en la barra de estado del ContainerWindow (ver figura 148).

Figura 148: método run de la clase HierarchicalBasic

```
public void run() {
      long timeIni = System.currentTimeMillis();
      //System.out.println("  "+this.clusterIcon.k);
      String distance = null;
      String measure;

      switch (clusterIcon.typeDistance){
         case 00:
             matrix = Distance.calculateDistanceMatrixEuclideanHamming(clusterIcon.dataset);
             distance = "Euclideans Hamming";
            break;
         case 01:
            matrix = Distance.calculateDistanceMatrixEuclideanProportion(clusterIcon.dataset);
            distance = "Euclideans Proportion";
            break;
         case 10:
            matrix = Distance.calculateDistanceMatrixMahattanHamming(clusterIcon.dataset);
            distance = "Manhattan Hamming";
            break;
         case 11:
            matrix = Distance.calculateDistanceMatrixManhattanProportion(clusterIcon.dataset);
            distance = "Manhattan Proportion";
            break;
         case 20:
            matrix = Distance.calculateDistanceMatrixChebyshevHamming(clusterIcon.dataset);
             distance = "Chebyshev Hamming";
            break;
         case 21:
            matrix = Distance.calculateDistanceMatrixChebyshevProportion(clusterIcon.dataset);
            distance = "Chebyshev Proportion";
            break;
         case 30:
            matrix = Distance.calculateDistanceMatrixMahalanobisHamming(clusterIcon.dataset);
            distance = "Mahalanobis Hamming";
            break;
         case 31:
            matrix = Distance.calculateDistanceMatrixMahalanobisProportion(clusterIcon.dataset);
            distance = "Mahalanobis Proportion";
            break;
      }
Continua…
```

```
      switch(clusterIcon.typeMeasure){
          case 1:
             this.algorithmSingle(this.clusterIcon.k);
             distance = " Hierarchical Simple "+ "distance "+distance;
             break;
          case 2:
             this.algorithmAverange(this.clusterIcon.k);
             distance = " Hierarchical Averange "+ "distance "+distance;
             break;
          case 3:
             this.algorithmComplete(this.clusterIcon.k);
             distance = " Hierarchical Complete "+ "distance "+distance;
             break;
       }

       long timeFin =System.currentTimeMillis() - timeIni;

        ContainerWindow.setStates("Time transcurrido Hierarchical "+distance+ ": "+timeFin+"
milisegundos" );

        this.clusterIcon.distancias = this.distancias.toArray();
        this.clusterIcon.assigCluster = this.assigCluster;
        this.clusterIcon.numElementsCluster = this.numElementsCluster;
        this.clusterIcon.meansCluster = this.meansCluster;
        this.clusterIcon.valuesAttributesCluster = this.valuesAttributesCluster;
        this.clusterIcon.nameAlgorithm = distance;

       if(clusterIcon.k>1)
         dataset.calculateDesviationStandand(this.clusterIcon);

       this.clusterIcon.nomImagen = this.clusterIcon.nameFigure;
       this.clusterIcon.repaint();

   }
```

La clase run es la encargada de calcular la matriz de distancia utilizando ejecutando los métodos de distancia. El cálculo de la matriz de distancia es independiente de los enlaces entre clúster definidos por el usuario. Para las diferentes variaciones de clustering jerárquico están los métodos:

- ✓ algorithmSingle(this.clusterIcon.k);
- ✓ algorithmAverange(this.clusterIcon.k);
- ✓ algorithmComplete(this.clusterIcon.k);

Los métodos de clustering jerárquico mencionados tienen los mismos pasos, la única diferencia está en la invocación del método para calcular la distancia entre

cluster (ver figura 149 en *). por eso solo se mostrara solo el método algorithmAverange.

Figura 149: método algorithmAverange.

```
private void algorithmAverange(int numCluster) {
        Vector activos;
        int i,j;

        activos = new Vector();
        Nodo auxNodo ;

        for(i=0; i< dataset.numInstances; i++ ){
           auxNodo = new Nodo();
           if(dataset.ini==0)
             auxNodo.name = "o"+(i);
           else{
              auxNodo.name = dataset.instancias[i][0].toString();
              if(auxNodo.name.length()>6 )
                 auxNodo.name = auxNodo.name.substring(0,6);
           }

           auxNodo.values =   dataset.instancias[i];
           auxNodo.nivel = 0;
           auxNodo.ind = i;
           activos.add(auxNodo);
        }

        double distance = 0;
        double distanciaMenor;

        nodoRaiz = null;
        Nodo nodo1, nodo2;
        int ind1=0, ind2 = 0;

        int iteraciones = 1;

        while (activos.size()!= numCluster && iteraciones != clusterIcon.numIte) {

           distanciaMenor = Double.MAX_VALUE;
           nodoRaiz = new Nodo();
           nodoRaiz.nivel = iteraciones;
           for (i = 0; i < activos.size() - 1; i++) {
              nodo1 = (Nodo) activos.get(i);
              for (j = i + 1; j < activos.size(); j++) {
                 nodo2 = (Nodo) activos.get(j);

                //* distancia Promedio o Averange
                 distance = this.calculateDistanceAverage(nodo1, nodo2,dataset.columnsNumeric);
continua…
```

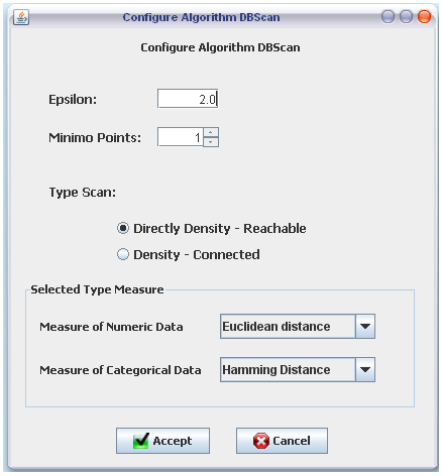
```
            if (distanciaMenor > distance) {
                    distanciaMenor = distance;
                    nodoRaiz.left = nodo1;
                    nodoRaiz.rigth = nodo2;
                    nodoRaiz.cantNodos = nodo1.cantNodos + nodo2.cantNodos;
                    nodoRaiz.name = nodo1.name + " + " + nodo2.name;
                    ind1 = i;
                    ind2 = j;
                }
            }
        }

        // reemplazo el valor del primer nodo unido con el nodo formado
        activos.set(ind1, nodoRaiz);
        // coloco nivel del padre
        nodoRaiz.left.nivelPadre = iteraciones;
        nodoRaiz.rigth.nivelPadre = iteraciones;
        // elimino el segundo nodo
        activos.remove(ind2);
        // numero de grupos
        nodoRaiz.kCluster = activos.size();
        iteraciones++;

    }
    nodoRaiz.nivelPadre = 1;
    printResult(nodoRaiz);
    this.clusterIcon.nodoRaiz = nodoRaiz;

    if(numCluster != 1 || clusterIcon.numIte != 0){
        this.clusterIcon.nodosGrupo = activos.toArray();
        this.clusterIcon.k = activos.size();
        calculateCentroidesCluster(activos);

    }
}
```

Paquete Density_Based

En este paquete se encuentran las clases FrmCOnfigureDBScan, DBScan y Point la primera clase permite configurar los parámetros de los algoritmos, la otra clase contiene los pasos de la técnica de clustering basado en densidad que a su vez hace uso de la clase Point.

Clase FrmConfigureDBScan: Permite al usuario establecer el área de barrido, presentando estas opciones con JRadioButton, también le permite escoger el tipo función de distancia a utilizar para los datos numéricos y también para el tipo de distancia entre atributos categóricos (ver figura 150). Presionando el botón aceptar, se establecen dichos parametros definidos, dentro de la instancia de la clase ClusteringIcon.

Figura 150: Instancia de la clase FrmConfigureDBScan.



Clase DBScan: Esta clase contiene métodos que realizan los pasos básicos de la técnica de clustering basada en densidad. Esta clase hereda de la clase Thread para crear un proceso independiente de la interfaz. El constructor de esta clase recibe como parámetro una instancia de la clase ClusteringIcon, (ver figura 151). Recibe la instancia de la clase ClusteringIcon porque ahí están los parámetros de configuración definidos por el usuario, además en esa instancia se va a colocar los resultados del algoritmo.

Figura 151: método constructor de la Clase DBScan.

```
public DBScan(ClusteringIcon clusterIcon){

      this.clusterIcon = clusterIcon;
      this.dataset = clusterIcon.dataset;
}
```

El algoritmo empieza invocando el método run (ver figura 152) que requiere dos parámetros. El primer parámetro es de tipo int y define la cantidad mínimo de puntos que debe haber en un grupo formado. El otro parámetro es de tipo double, que es el límite de distancia máximo entre el núcleo y un punto. Aquí se invoca el método que ha según los parámetros establecidos por el usuario en cuanto a las funciones de distancia y la forma de establecer el área de barrido. Al finalizar cualquiera de los dos métodos, se asigna a la instancia de la clase ClusteringIcon los valores medios de los clústeres formados y el número de elementos asignados en cada grupo. Además el método run es el encargado de tomar el tiempo de ejecución del algoritmo, tomando el tiempo del sistema con el método estático System.currentTimeMillis que retorna una variable tipo long con el valor del tiempo en milisegundos del sistema. Para saber qué tiempo tardo, se guarda el tiempo inicial, cuando termine el método llamado se resta con el tiempo final y

se muestra en la barra de estado del ContainerWindow. (Ver figura 152) los métodos que se encuentran implementados en esta clase son las diferentes combinaciones entre las funciones de distancia para los atributos numéricos y categóricos y la forma de establecer el área de barrido ya sea directa o conectada.

Figura 152: método run de la clase DBScan.

```
public void run() {
      long timeIni = System.currentTimeMillis();
      int minPoints = clusterIcon.minPoints;
      double epsilon = clusterIcon.epsilon;
      switch (clusterIcon.typeDistance){

         case 00:
           switch(clusterIcon.typeMeasure){
           case 1:     this.algorithmDirectlyEuclideanHamming( minPoints,epsilon);
              break;
           case 2:     this.algorithmConnectedEuclideanHamming(minPoints,epsilon );
              break;
          }
         break;

         case 01:
           switch(clusterIcon.typeMeasure){
           case 1:    this.algorithmDirectlyEuclideanProportion( minPoints,epsilon);
              break;
           case 2:    this.algorithmConnectedEuclideanProportion(minPoints,epsilon );
              break;
          }
         break;

         case 10:
           switch(clusterIcon.typeMeasure){
           case 1:     this.algorithmDirectlyManhattanHamming( minPoints,epsilon);
              break;
           case 2:     this.algorithmConnectedManhattanHamming(minPoints,epsilon );
              break;
          }
         break;

         case 11:
           switch(clusterIcon.typeMeasure){
           case 1:     this.algorithmDirectlyManhattanProportion( minPoints,epsilon);
              break;
           case 2:     this.algorithmConnectedManhattanProportion(minPoints,epsilon );
              break;
          }
         break;
continua…
```

```
                case 20:
                    switch(clusterIcon.typeMeasure){
                    case 1:   this.algorithmDirectlyChebyshevHamming( minPoints,epsilon);
                        break;
                    case 2:    this.algorithmConnectedChebyshevHamming(minPoints,epsilon );
                               break;
                     }
                  break;


                case 21:
                  switch(clusterIcon.typeMeasure){
                  case 1:    this.algorithmDirectlyChebyshevProportion( minPoints,epsilon);
                     break;
                  case 2:    this.algorithmConnectedChebyshevProportion(minPoints,epsilon );
                     break;
                 }
                break;

                 case 30:
                  switch(clusterIcon.typeMeasure){
                  case 1:    this.algorithmDirectlyMahalanobisHamming( minPoints,epsilon);
                     break;
                  case 2:    this.algorithmConnectedMahalanobisHamming(minPoints,epsilon );
                     break;
                 }
                break;

                case 31:
                  switch(clusterIcon.typeMeasure){
                  case 1:   this.algorithmDirectlyMahalanobisProportion( minPoints,epsilon);
                     break;
                  case 2:  this.algorithmConnectedMahalanobisProportion(minPoints,epsilon );
                     break;
                 }
                break;

            }
            long timeFin =System.currentTimeMillis() - timeIni;
            ContainerWindow.setStates("Time transcurrido DBScan: "+timeFin+"
              milisegundos" );

            this.clusterIcon.k = this.meansCluster.length;
            this.clusterIcon.assigCluster = this.assigCluster;
            this.clusterIcon.numElementsCluster = this.numElementsCluster;
```

Los métodos basados en densidad tiene unos pasos básicos (ver figura 153) la única diferencia que tienen entre es que los métodos directos y hacen en una iteración la comparación de un solo CorePoint (ver figura 154), en cambio el conectado (ver figura 155) es un método recursivo que cuando adiciona un punto

a un cluster lo convierte en CorePoint para compararlo con los puntos restantes lo que hace de este método un método recursivo.

Figura 153: Método básico algorithmDBScan.

```
public void algorithmDirectlyEuclideanHamming(int minPoints,double epsilon){

    int iteration;
    Vector pila;
    Vector arrayCorePoints;
    boolean converge = false;
    int i,j;
    pila = new Vector();
    arrayCorePoints = new Vector();
    this.assigCluster = new int[dataset.numInstances];

    //1. colocar todos los registros en una vector
    for(i=0; i < dataset.numInstances; i++){
       pila.add(new Point(("o"+(i+1) ) , dataset.instancias[i],i) );
       this.assigCluster[i] = -1;
    }

    int numRandom;
    Point corePoint;

    iteration = 1;
    //2. mientras no se llegue a la convergencia si la cantidad de valores en el vector son menores
         a  la cantidad minima de puntos, se llego a un punto de convergencia

    while(pila.size() >= minPoints){
       //2.1 elegir un nucleo del vector r de forma aletoria
       numRandom = (int) Math.random()*pila.size();
       corePoint  =  (Point) pila.get( numRandom );
       // eliminarlo del vector
       pila.remove(numRandom);
       //calculate density entre el nucleo y los valores del vector
       // y crear grupo con los que la distancia sea menor a un epsilon
         this.directlyDensityReachableEuclideanHamming(corePoint,pila,dataset,epsilon,
              minPoints);

       //mirar si el numero de puntos mayor o igual que el valor de puntos minimos
        // considerarlo como punto nucleo y almacenarlo
       if(corePoint.points.size()>= (minPoints) ){
          arrayCorePoints.add(corePoint);
       }
      iteration++;
    }
    this.printResult(arrayCorePoints);
    createMeansCluster(arrayCorePoints);
  }
```

Figura 154: Pasos metodos directlyDensityReachable.

```
private void directlyDensityReachable(Point corePoint,Vector pila,DataSet dataset, double
epsilon,int minPoints){
    int i;
    Point point;
    double distance;
    Vector  auxPoints = new Vector();

    corePoint.sumDistance = 0;

    for(i=0; i< pila.size();i++){
            point = (Point) pila.get(i);
            distance = Distance.measureDistanceEuclideanHamming(corePoint.values,point.values,
                        dataset.columnsNumeric,dataset.ini

            if(distance <= epsilon){
               auxPoints.add( point);
               pila.remove(i);
               corePoint.sumDistance+= distance;
               i--;
            }
     }
    corePoint.points = auxPoints;
   }
```

Figura 155: Pasos metodos densityConnected.

```
private void densityConnected (Point corePoint,Vector pila,
                DataSet dataset, double epsilon,int minPoints,double sumDistance,Vector addPoints){
    int i;
    Point point;
    double distance;

    for(i=0; i< pila.size();i++){
            point = (Point) pila.get(i);
            distance = Distance.measureDistanceEuclideanProportion(corePoint.values,point.values,
                    dataset.columnsNumeric,dataset.ini,dataset.atriCategorical);
            if(distance <= epsilon){
               addPoints.add( point);
               pila.remove(i);
               // conected
               sumDistance += distance;
               densityConnected (point,pila,dataset,epsilon,minPoints,sumDistance,addPoints);
               sumDistance += point.sumDistance;
               i--;
            }
     }

    corePoint.sumDistance=sumDistance;
```

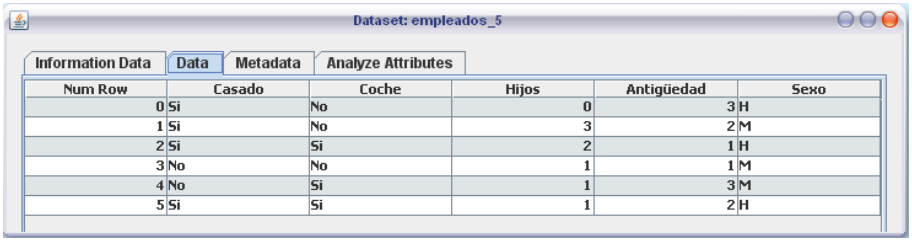
8. PRUEBAS Y ANALISIS DE RESULTADOS

Las pruebas presentadas a continuación se desarrollaron utilizando un computador Intel® Core™ 2 CPU 4400 @ 2GHZ, Disco Duro de 250 GB, memoria RAM de 3 Gigas y tarjeta de video de 64MB, sistema operativo Windows XP SP2 y la versión jdk. 1.6 de java.

8.1 PRUEBAS DE VALIDACION

El objetivo de estas pruebas fue validar la correcta implementación de los algoritmos, con respecto a las pruebas manuales desarrolladas en el capítulo 3. Para ello se muestra en la figura 156 el conjunto de datos empleados que fue cargado desde un archivo plano denominado empleadosBase.arff y el cual tiene los mismos valores presentados en la tabla 1. Datos que sirvieron para realizar todos los ejemplos.

Figura 156: Vista del conjunto de datos empleados en la ventana ViewData.

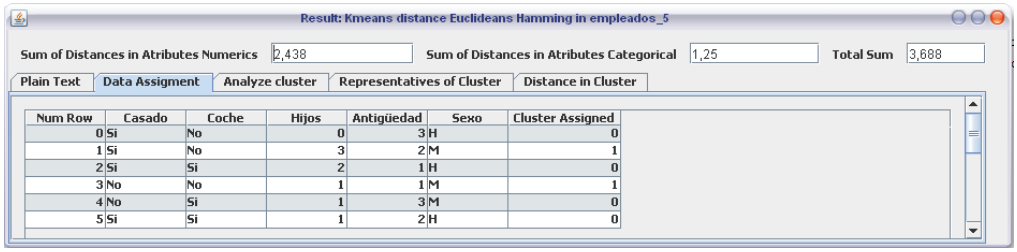


Num Row	Casado	Coche	Hijos	Antigüedad	Sexo
0	Si	No	0	3	H
1	Si	No	3	2	M
2	Si	Si	2	1	H
3	No	No	1	1	M
4	No	Si	1	3	M
5	Si	Si	1	2	H

8.1.1 Prueba de validación algoritmo kmeans.

Los parámetros utilizados aquí son: selección de semillas tipo McQueen, numero de clústeres a formar 2 (k=2), el método de parada es la convergencia y se selecciona el tipo de distancia para datos numéricos la distancia euclidiana y para los atributos categóricos la distancia Hamming. Contrastando la tabla 8 que presenta el resultado del ejemplo 1, con el presentado por Rasemus en la figura 157, coinciden los valores de asignación de clúster, que certifica la correcta implementación del algoritmo kmeans en Rasemus.

Figura 157: Resultado del algoritmo kmeans en Rasemus.

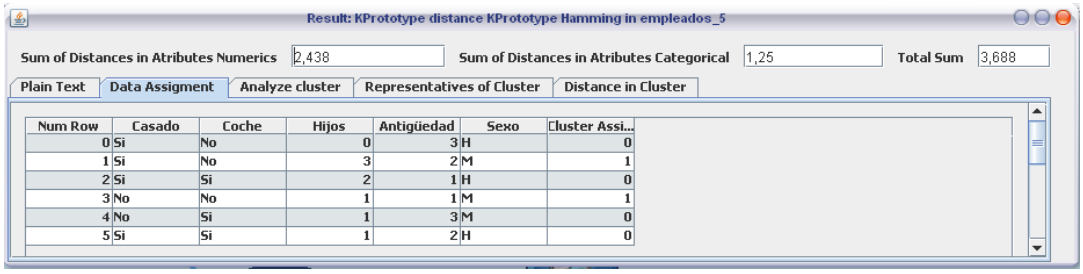


Num Row	Casado	Coche	Hijos	Antigüedad	Sexo	Cluster Assigned
0	Si	No	0	3	H	0
1	Si	No	3	2	M	1
2	Si	Si	2	1	H	0
3	No	No	1	1	M	1
4	No	Si	1	3	M	0
5	Si	Si	1	2	H	0

8.1.2 Prueba de validación algoritmo KPrototype.

Los parámetros utilizados aquí son: número de clústeres a formar 2 (k=2), el método de parada es la convergencia y se selecciona el tipo de distancia para datos numéricos la distancia euclidiana y para los atributos categóricos la distancia Hamming. Hay que tener en cuenta que como el algoritmo kprototype hace una asignación aleatoria al inicio del proceso, los resultado pueden variar para este caso se hizo coinciden dicha asignación. Contrastando la tabla 13 que presenta el resultado del ejemplo 2, con el presentado por Rasemus en la figura 158, coinciden los valores de asignación de clúster, que certifica la correcta implementación del algoritmo kprototype en Rasemus.

Figura 158: Resultado del algoritmo kprototype en Rasemus.

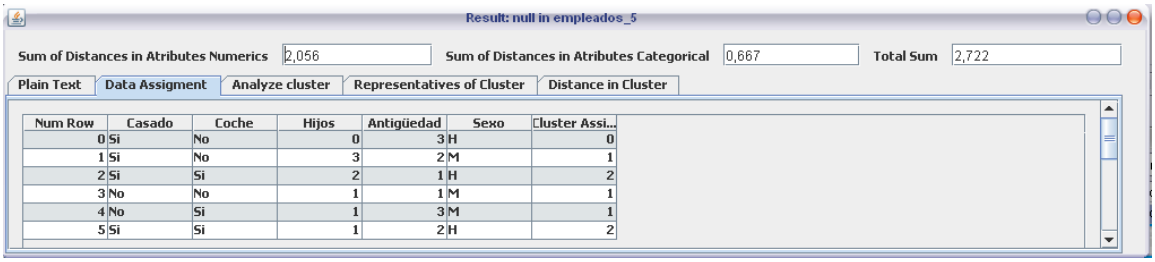


Num Row	Casado	Coche	Hijos	Antigüedad	Sexo	Cluster Assi...
0	Si	No	0	3	H	0
1	Si	No	3	2	M	1
2	Si	Si	2	1	H	0
3	No	No	1	1	M	1
4	No	Si	1	3	M	0
5	Si	Si	1	2	H	0

8.1.3 Prueba de validación algoritmo WayCluster.

Los parámetros utilizados aquí son seleccionadas asi: el tipo de distancia para datos numéricos la distancia euclidiana y para los atributos categóricos la distancia Hamming. Contrastando la tabla 27 que presenta el resultado del ejemplo 3, con el presentado por Rasemus en la figura 159, coinciden los valores de asignación de clúster, que certifica la correcta implementación del algoritmo wayCluster en Rasemus.

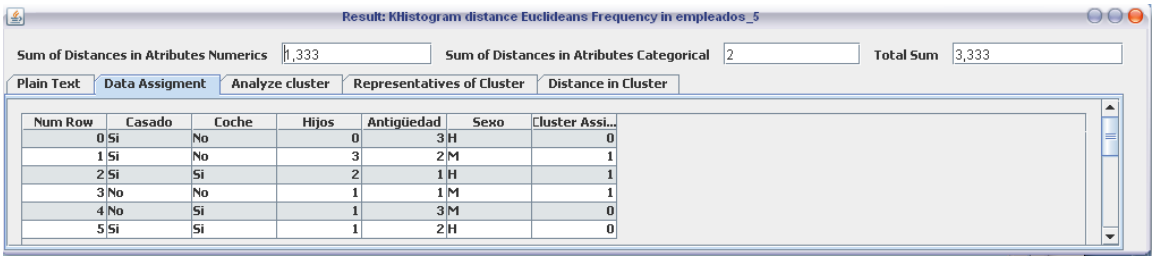
Figura 159: Resultado del algoritmo WayCluster en Rasemus.



8.1.4 Prueba de validación algoritmo KFrequency.

Los parámetros utilizados aquí son: número de clústeres a formar 2 (k=2), el método de parada es la convergencia y se selecciona solo el tipo de distancia para datos numéricos, la distancia euclidiana. Contrastando la tabla 32 que presenta el resultado del ejemplo 4, con el presentado por Rasemus en la figura 160, coinciden los valores de asignación de clúster, que certifica la correcta implementación del algoritmo kfrequency en Rasemus.

Figura 160: Resultado del algoritmo kfrequency en Rasemus.



8.1.5 Prueba de validación algoritmos jerárquicos.

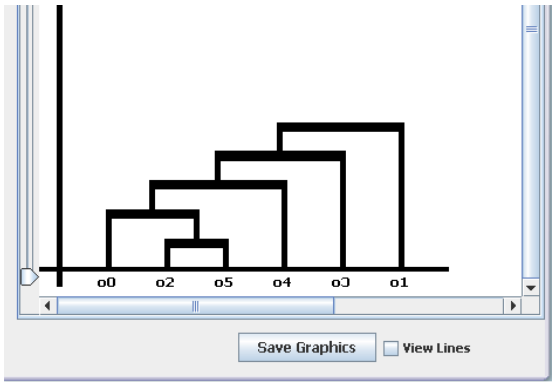
Los parámetros utilizados fueron: el tipo de enlace entre grupos, se fue variando, el tipo de parada, fue formar todo el dendograma y se selecciono el tipo de distancia, para datos numéricos, la distancia euclidiana y para los atributos categóricos, la distancia Hamming. Aquí el resultado se constato con los dendogramas resultados en los ejemplos presentados en el capítulo 3 y los presentados por la herramienta.

Prueba de validación algoritmos jerárquicos con enlace Mínimo.

Contrastando la figura 35, que presenta el dendograma resultante del ejemplo 5, con el presentado por Rasemus en la figura 161, coinciden los dendogramas, que

certifica la correcta implementación del algoritmo de clustering jerárquico con enlace mínimo en Rasemus.

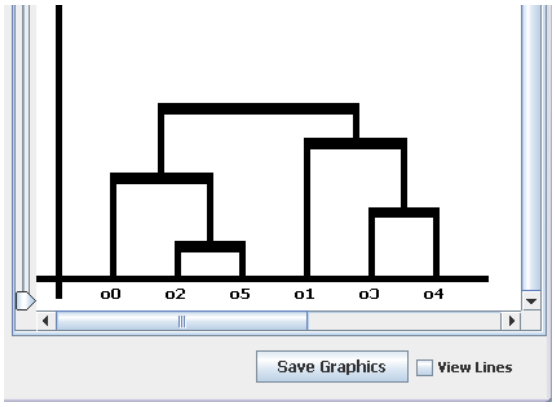
Figura 161: Resultado del algoritmo jerárquico con enlace Mínimo en Rasemus.



Prueba de validación algoritmo jerárquico con enlace máximo.

Contrastando la figura 41, que presenta el dendograma resultante del ejemplo 6, con el presentado por Rasemus en la figura 162, coinciden los dedogramas, que certifica la correcta implementación del algoritmo de clustering jerárquico con enlace máximo en Rasemus.

Figura 162: Resultado del algoritmo jerárquico con enlace máximo en Rasemus.

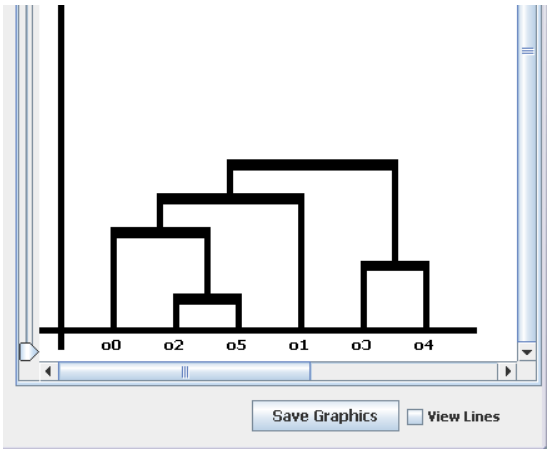


Prueba de validación algoritmo jerárquico con enlace Promedio

Contrastando la figura 47, que presenta el dendograma resultante del ejemplo 7, con el presentado por Rasemus en la figura 163, coinciden los dendogramas, que

certifica la correcta implementación del algoritmo de clustering jerárquico con enlace promedio en Rasemus.

Figura 163: Resultado del algoritmo jerárquico con enlace promedio en Rasemus.



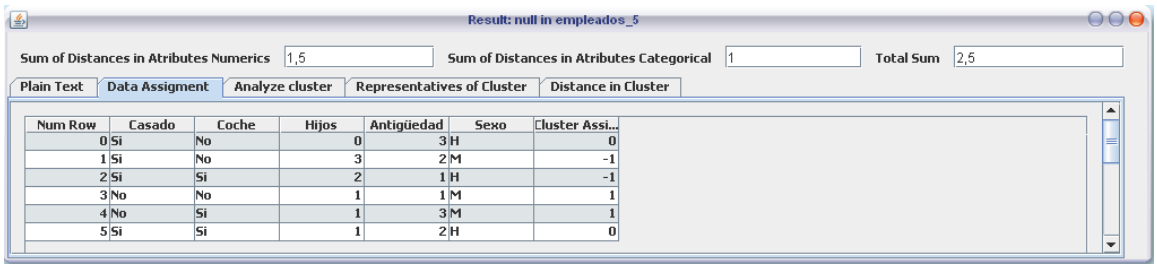
8.1.6 Prueba de validación algoritmo DBScan.

Los parámetros utilizados aquí son valor mínimo de puntos igual a 1 y épsilon de 3.0 y se selecciona solo el tipo de distancia para datos numéricos, la distancia euclidiana y para categóricos la distancia hamming, y se vario el tipo de barrido.

Prueba de validación algoritmo DBScan con barrido directo.

Contrastando la tabla 48 que presenta el resultado del ejemplo 8, con el presentado por Rasemus en la figura 164, coinciden los valores de asignación de clúster, que certifica la correcta implementación del algoritmo DBScan con barrido directo en Rasemus. hay que tener en cuenta que la asignación en el clúster -1 significa que es ruido o noise.

Figura 164: Resultado del algoritmo DBScan con barrido directo en Rasemus.

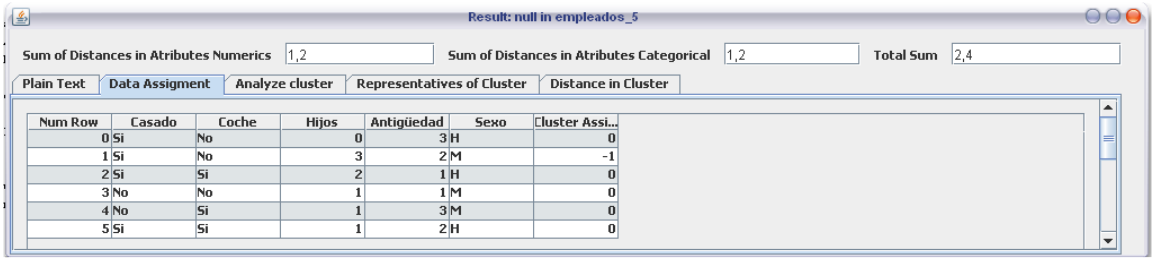


Num Row	Casado	Coche	Hijos	Antigüedad	Sexo	Cluster Assi...
0	Si	No	0	3	H	0
1	Si	No	3	2	M	-1
2	Si	Si	2	1	H	-1
3	No	No	1	1	M	1
4	No	Si	1	3	M	1
5	Si	Si	1	2	H	0

Prueba de validación del algoritmo DBScan con barrido conectado.

Contrastando la tabla 53 que presenta el resultado del ejemplo 9, con el presentado por Rasemus en la figura 165, coinciden los valores de asignación de clúster, que certifica la correcta implementación del algoritmo DBScan con barrido conectado en Rasemus.

Figura 165: Resultado del algoritmo DBScan con barrido conectado en Rasemus.



Num Row	Casado	Coche	Hijos	Antigüedad	Sexo	Cluster Assi...
0	Si	No	0	3	H	0
1	Si	No	3	2	M	-1
2	Si	Si	2	1	H	0
3	No	No	1	1	M	0
4	No	Si	1	3	M	0
5	Si	Si	1	2	H	0

Con las pruebas anteriores se verifico el correcto funcionamiento de los algoritmos implementados.

8.2 PRUEBAS DE FUNCIONALIDAD CON ATRIBUTOS NUMERICOS

El objetivo de estas pruebas es mostrar la funcionalidad de los algoritmos implementados en Rasemus con repositorios reales que tengan todos los atributos de tipo numérico. Para ellos se utilizara el conjunto de datos formado por la UNESCO en 1992 denominado MUNDODES.arff [57], para evaluar el desarrollo de 91 países. Este conjunto de datos consta de 91 registros y 6 atributos, atributos que corresponden a indicadores de desarrollo. Estos son:

- ✓ Tasa Nat.: Ratio de natalidad por 1000 habitantes.
- ✓ Tasa Mort: Ratio de mortalidad por 1000 habitantes.
- ✓ Mort.Inf: Mortalidad infantil (por debajo de un año).
- ✓ Esp.Hom: Esperanza de vida en hombres.
- ✓ Esp.Muj: Esperanza de vida en mujeres.
- ✓ PNB: Producto Nacional Bruto per cápita.

En la tabla 55 se presentan las características de los atributos de MUNDODES.arff.

Tabla 55: Característica de los atributos del conjunto de datos MUNDODES

	Tasa_Na	Tasa_Mort	Mort_Inf	Esp.Hom	Esp.Muj	PNB
Mínimo	9.7	2.2	4.5	38.1	41.2	80.0
Media	29.460	10.734	55.281	61.381	66.031	5741.252
Máximo	52.2	25.0	181.6	75.9	81.8	34064.0
Desviación Media	12.060	3.546	40.083	8.049	9.361	6413.878
Desviación Estándar	13.670	4.684	46.302	9.728	11.131	8093.680
Varianza	187.666	21.938	2143.905	94.628	123.892	6550765.357
Suma de Error Cuadrático	16889.938	1974.424	192951.458	8516.518	11150.274	58956888.212

En las figuras 166, 167, 168,169 y 170 se presenta los diagramas de dispersión de los atributos, en ellas se puede observar que sus tendencia son diferentes, que no hay a simple vista un único patrón de comportamiento entre todos ellos.

Figura 166: Graficas de dispersión del atributo Tasa_Mort con respecto a los atributos: a) Mort_Inf, b) Esp.Hom, c) Esp.Muj, d) PNB.

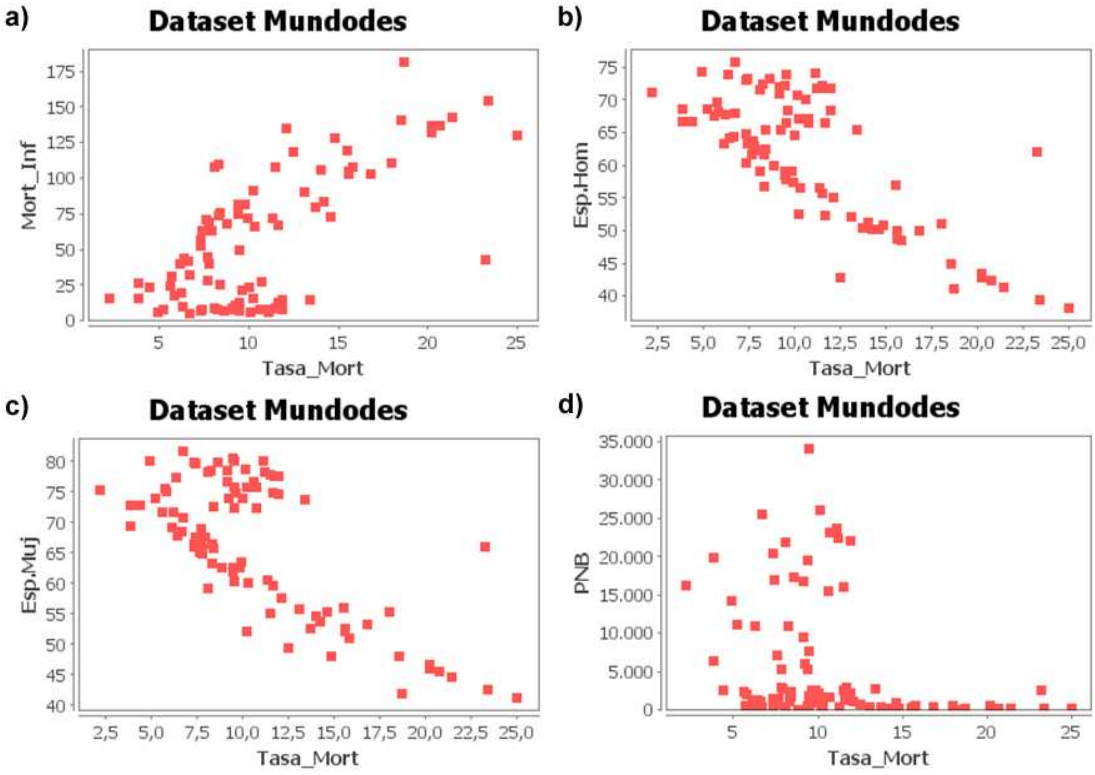


Figura 167: Graficas de dispersión del atributo Tasa_Na con respecto a los atributos: a) Tasa_Mort, b) Mort_Inf, c) Esp.Hom, d) Esp.Muj, e) PNB.

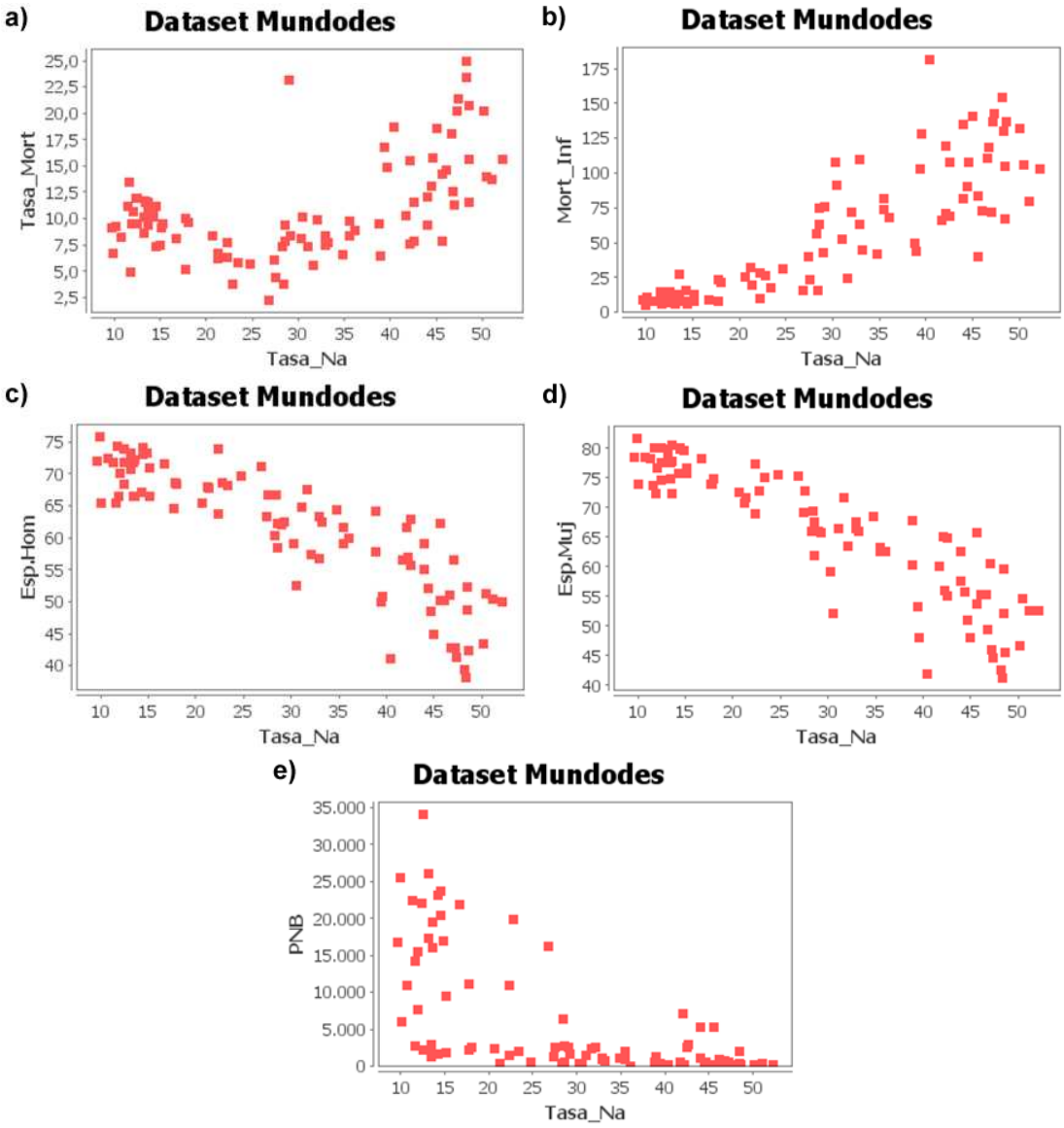


Figura 168: Graficas de dispersión del atributo Mort_Inf con respecto a los atributos: a) Esp.Hom, b) Esp.Muj, c) PNB.

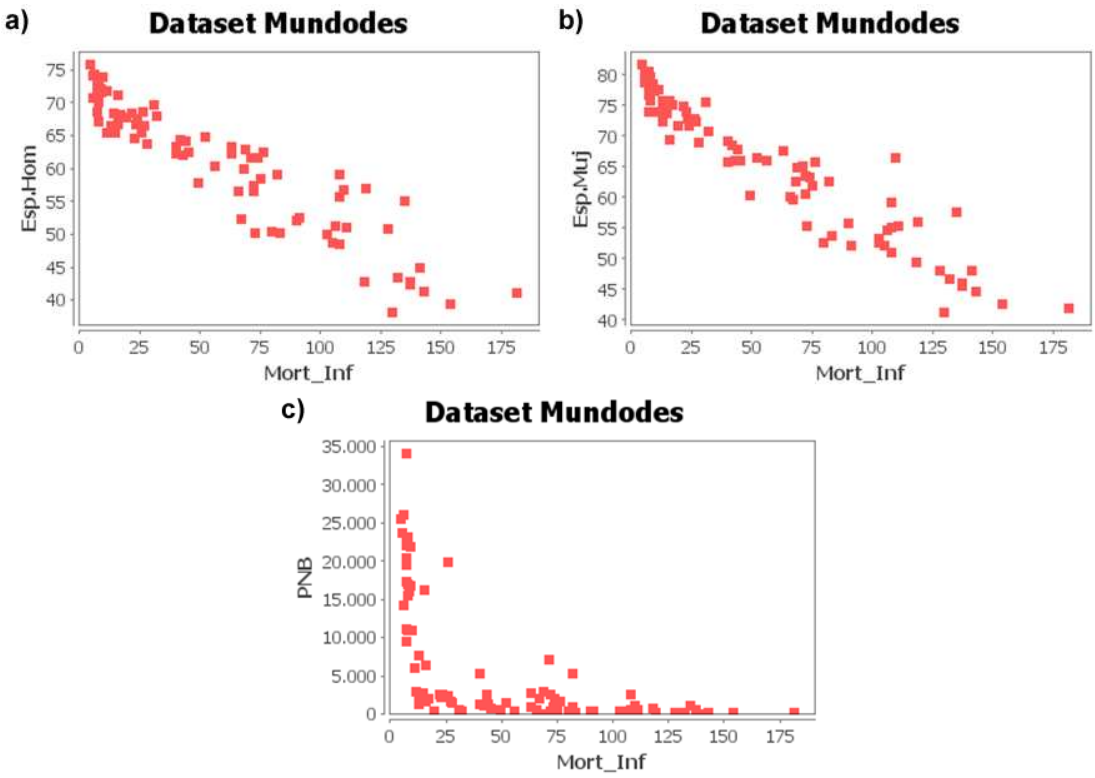


Figura 169: Graficas de dispersión del atributo Esp.Hom con respecto a los atributos: a) Esp.Muj, b) PNB.

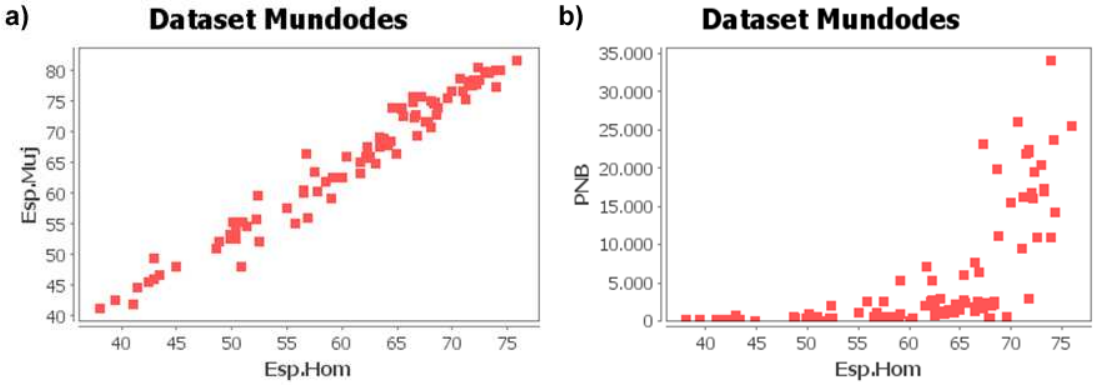
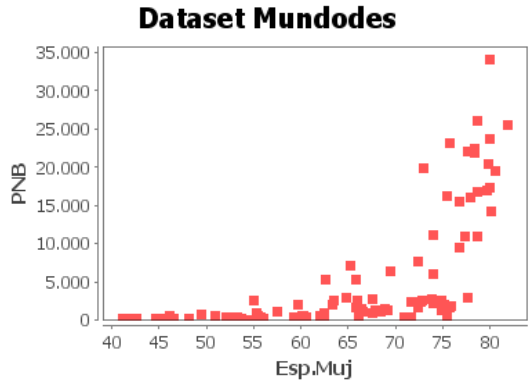


Figura 170: Grafica de dispersión del atributo Esp.Muj con respecto a PNB.



8.2.1 Calidad de clúster formados con los algoritmos de clustering particional utilizando la distancia euclidiana.

En la siguiente prueba se evalúa la calidad de los clústeres formados. La prueba consistió en variar el numero de clústeres a formar (K). Primero se muestra el comportamiento de la suma del promedio del error cuadrático de los atributos en cada clúster (SPECAC) en la tabla 56, esto con el fin de determinar el comportamiento de la SPECA para observar su efecto en la dispersión dentro de los clústeres mostrando la grafica la grafica de comportamiento de la SPECAC con la variación de k en la figura 171, también se presentan en la tabla 57 los valores del promedio de la distancia euclidiana entre los centros de clústeres para determinar el comportamiento de la semejanza entre ellos, es decir que tan separados están los clústeres, para ello también se realizo la figura 172, en donde se observa la tendencia de la SPECAC.

Tabla 56: SPECAC de los algoritmos particionales variando el valor de k utilizando la distancia euclidiana.

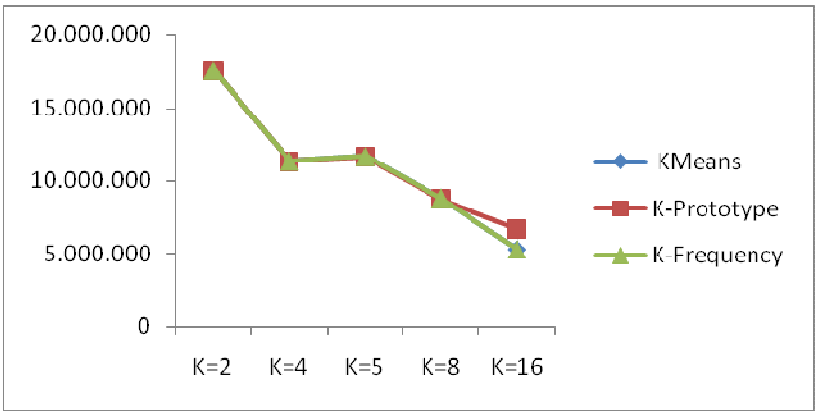
Valor de k	Algoritmos		
	KMeans	K-Prototype	K-Frequency
K=2	34223685,510	34223685,51	34223685,510
K=4	28915660,126	28915660,126	28915660,126
K=5	23684386,428	23684386,428	23684386,428
K=8	23483770,646	23471445,172	23483770,646
K=16	23382509,861	18792209,501	23382509,861

Figura 171: SPECAC de los algoritmos particionales, variando k utilizando la distancia euclidiana, aplicados a MUNDODES.

Tabla 57: Promedio de distancia entre los centros de clúster formados por los algoritmos particionales variando el valor de k, utilizando la distancia euclidiana.

Valor de k	Algoritmos		
	KMeans	K-Prototype	K-Frequency
K=2	17581.556	17581.556	17581.556
K=4	11344.322	11344.322	11344.322
K=5	11660.458	11660.458	11660.458
K=8	8790.567	8759.389	8790.567
K=16	5284.424	6689.291	5284.424

Figura 172: Promedio de distancia entre los centros de clúster formados por los algoritmos particionales, variando k, utilizando la distancia euclidiana.



Analizando los resultados de las pruebas antes mencionadas, se determina que al aumentar el número de clústeres, se disminuye la suma de promedio del error

cuadrático de clúster SPECAC, lo que indica que los elementos son más similares cuando el valor de k aumenta.

Para este caso en particular en la figura 171 se observa que después del valor de k = 5, los valores de la ESPECAC es relativamente bajo comparado con la diferencia entre K=4 y k=5 y k=5 y k=8 que significa un posible número ideal de clústeres a formar.

Analizando los datos de la tabla 57, en donde se encuentran los resultados de la distancia promedio entre los centros de grupo formados al ir variando el valor de k, se puede establecer que la distancia entre ellos disminuye al incrementar el valor de k (ver figura 172), lo que implica que entre menor número de clústeres, los centros de clúster están más separados.

8.2.2 Funcionalidad de los algoritmos de clustering particional con k=5 utilizando la distancia euclidiana.

El objetivo de esta prueba es hacer análisis de las características de los representantes de cada clúster resultantes con el algoritmo kmeans utilizando la distancia euclidiana solo se presentan los resultados del algoritmo kmeans debido a que con k=5 todos los algoritmos de tipo particional obtuvieron el mismo resultado que se encuentran en la tabla 58 y se presentan los datos de la suma de los promedios del error cuadrático de los atributos en cada clúster (SPECAC) que se presentan en la tabla 59.

Tabla 58: Representantes de cada clúster formados con k=5 en el algoritmo kmeans utilizando la distancia euclidiana aplicado a MUNDODES

Cluster	Tasa_Na	Tasa_Mort	Mort_Inf	Esp.Hom	Esp.Muj	PNB	Por.
Cluster 0	39,37	12,97	90,82	53,87	57,28	531,90	45,05%
Cluster 1	28,17	8,06	34,37	64,67	69,44	6722,86	7,69%
Cluster 2	14,16	9,07	8,67	71,9	78,53	23484,00	12,09%
Cluster 3	15,27	7,4	8,72	72,14	77,83	14625,00	10,99%
Cluster 4	25,5	9,76	40,16	64,19	69,65	2227,82	24,18%

Tabla 59: SPECAC en cada clústeres formados con k=5 en el algoritmo kmeans utilizando la distancia euclidiana aplicado a MUNDODES

Cluster	Tasa_Na	Tasa_Mort	Mort_Inf	Esp.Hom	Esp.Muj	PNB	Total Sum
Cluster 0	88,33	27,02	1595,37	72,42	85,91	130787,70	132656,75
Cluster 1	215,79	3,53	814,30	13,49	22,90	1972506,12	1973576,13
Cluster 2	10,38	5,29	31,41	5,57	5,55	15162298,18	15162356,38
Cluster 3	27,13	7,14	6,10	2,79	3,56	6200565,00	6200611,71
Cluster 4	109,03	13,41	726,12	19,26	34,03	214283,60	215185,46
Total	450,66	56,37	3173,31	113,54	151,94	23680440,61	23684386,43

Las ver figuras 173, 174, 175, 176 y 177 presentan los diagramas de dispersión, ilustrando los clústeres formados.

Figura 173: Graficas de dispersión mostrando los clústeres formados por kmeans con k=5 utilizando la distancia euclidiana del atributo Tasa_Na con respecto a los atributos: a) Tasa_Mort, b) Mort_Inf, c) Esp.Hom, d) Esp.Muj, e) PNB.

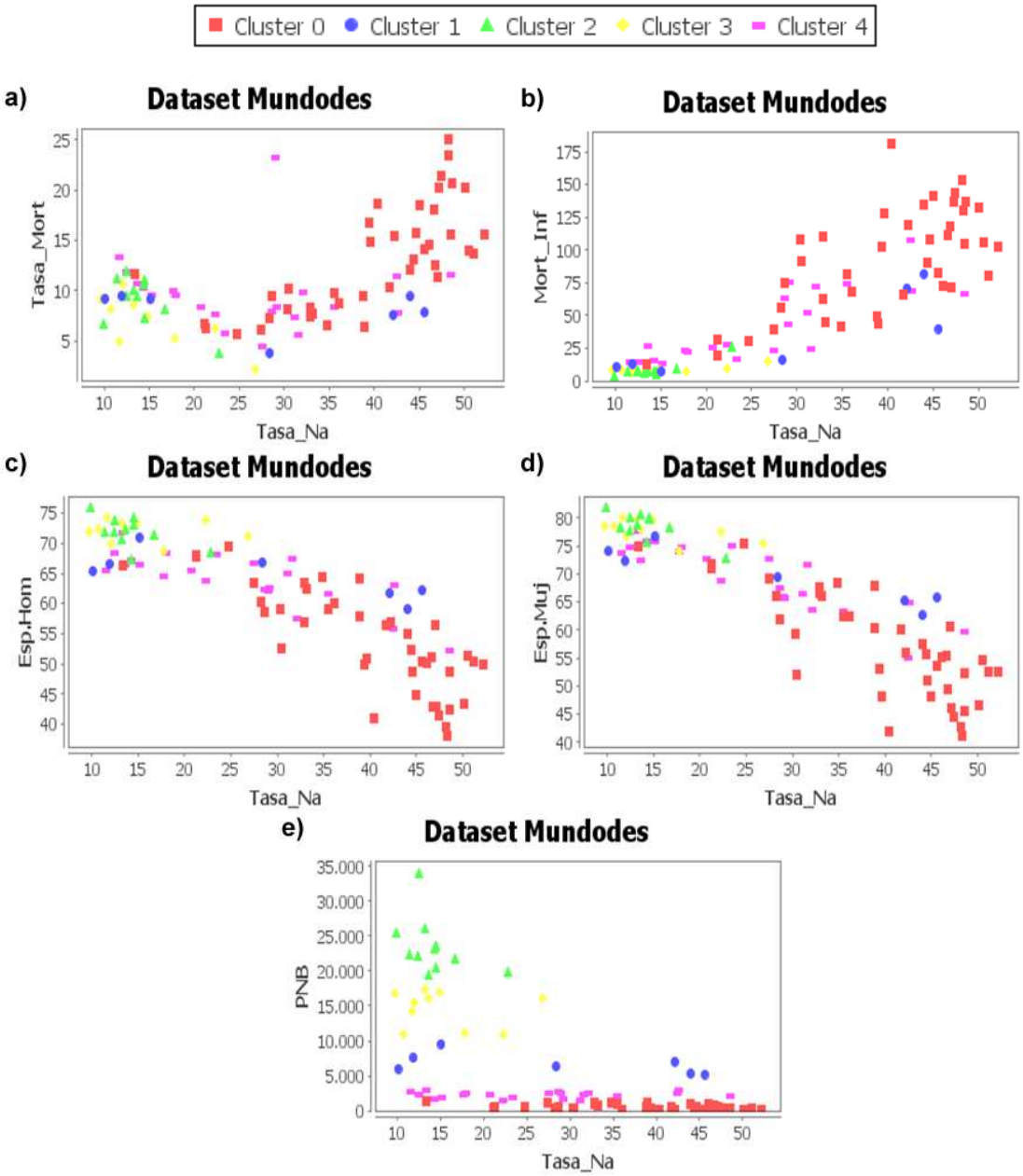


Figura 174: Graficas de dispersión mostrando los clústeres formados por kmeans con k=5 utilizando la distancia euclidiana del atributo Tasa_Mort con respecto a los atributos: a) Mort_Inf, b) Esp.Hom, c) Esp.Muj, d) PNB.

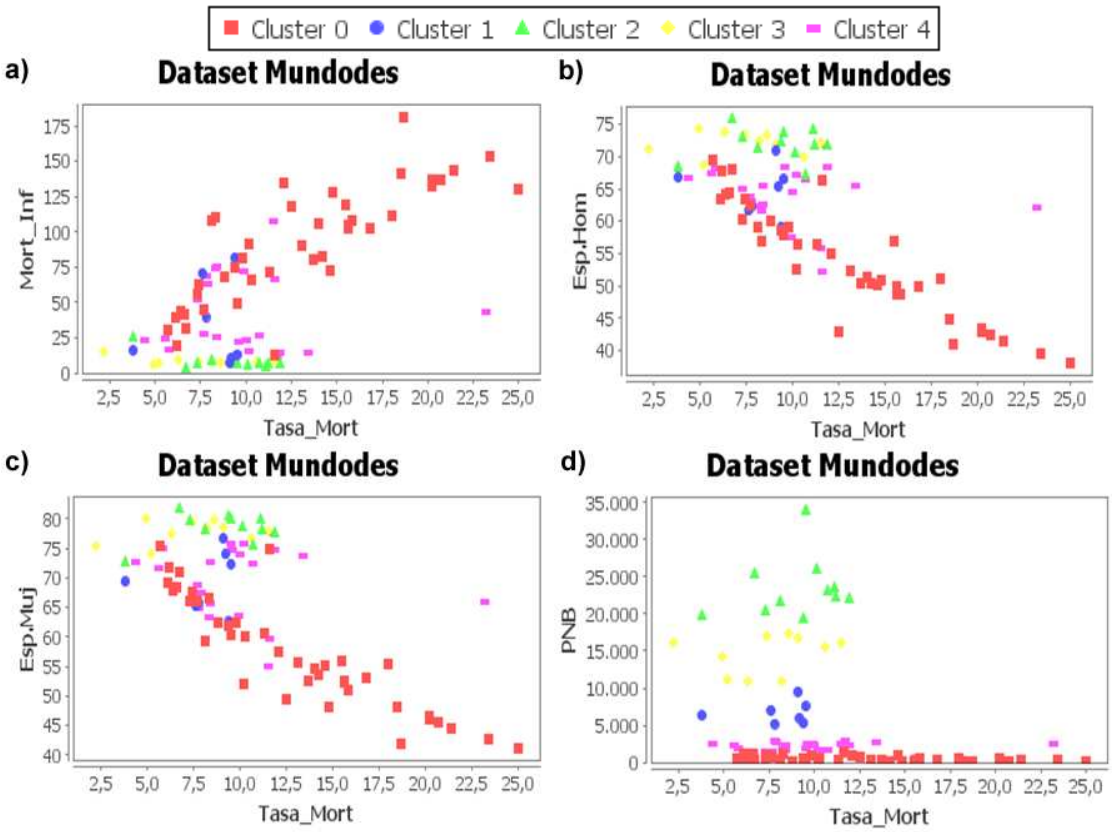
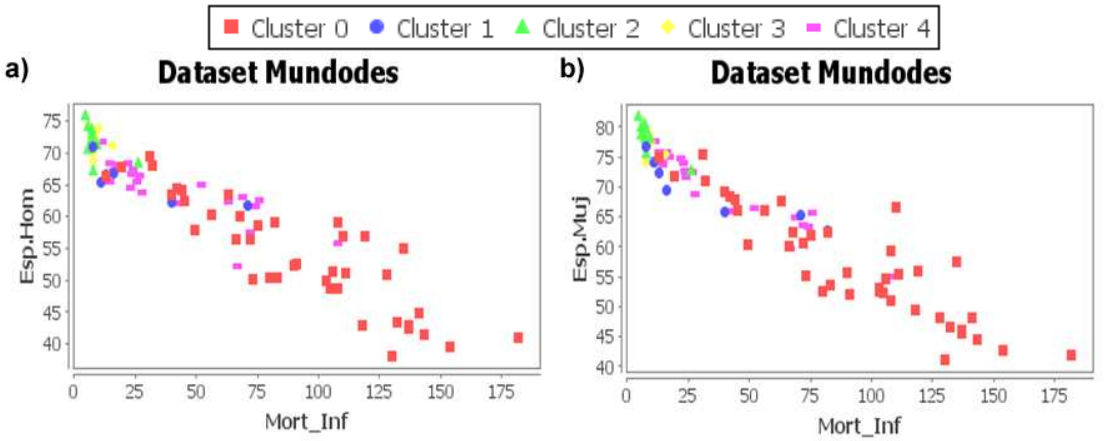


Figura 175: Graficas de dispersión mostrando los clústeres formados por kmeans con k=5 utilizando la distancia euclidiana del atributo Mort_Inf con respecto a los atributos: a) Esp.Hom, b) Esp.Muj, c) PNB.



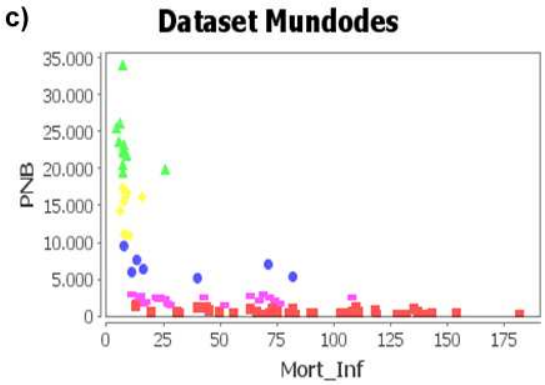


Figura 176: Graficas de dispersión mostrando los clústeres formados por kmeans con k=5 utilizando la distancia euclidiana del atributo Esp.Hom con respecto a los atributos: a) Esp.Muj, b) PNB.

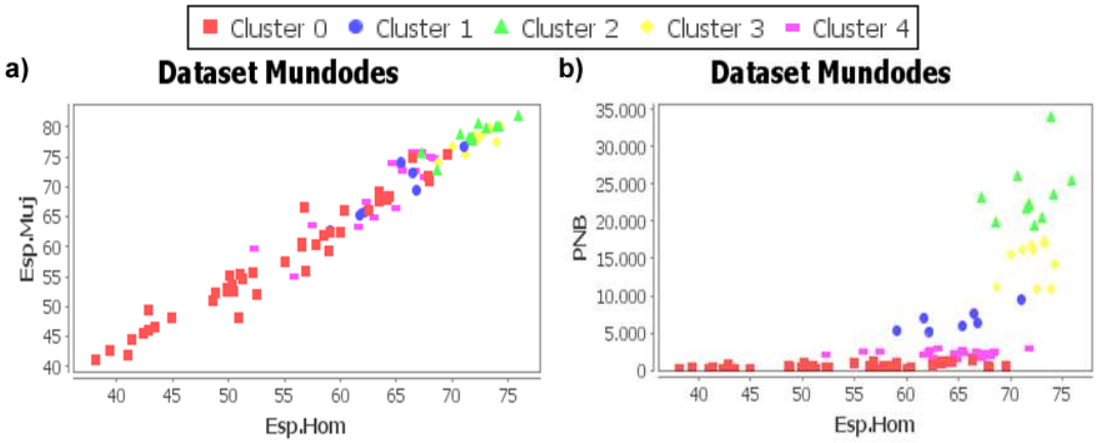
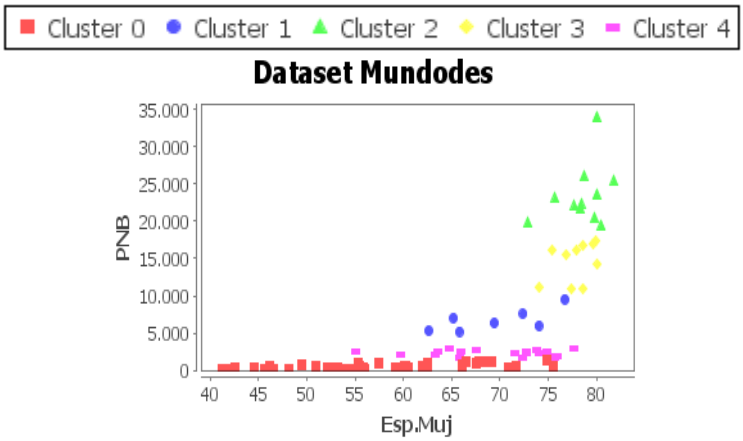


Figura 177: Graficas de dispersión mostrando los clústeres formados por kmeans con k=5 utilizando la distancia euclidiana del atributo Esp.Hom con respecto a el atributo PNB.



Analizando los representantes de los clústeres presentados en la tabla 58, se deduce que el clúster 0 está formado por el 45,05% de los países estudiados. Estos son los países que tienen un menor desarrollo, teniendo un valor promedio de 531,90 de producto interno bruto per cápita, con una mortalidad infantil muy alta y teniendo una expectativa de vida inferior a todos los demás clústeres formados tanto en hombres como en mujeres. Caso contrario es el clúster 2 al cual pertenece el 12,09% de los países, que son los países tienen entre ellos un promedio de 23484 producto interno bruto per cápita, una promedio de natalidad de 14,16 que es el más bajo comparado con el resto de clústeres y teniendo la esperanza de vida tanto de hombre como de mujeres más alta de todos los clústeres. Por otra parte se puede establecer que el atributo que más incidencia en la división de los clústeres es el atributo PNB, debido a sus valores elevados con respecto al resto de atributos, teniendo un valor representativo en la SPECAC, esto se puede observar en la tabla 59. Esto también se puede observar en las figuras 173, 174, 175, 176 y 177 que corresponden a los diagramas de dispersión de los atributos del conjunto de datos, y presenta que las figuras que tienen representados los valores del atributo PNB están bien definimos los clústeres formados.

8.2.3 Calidad de clúster formados con los algoritmos de tipo particional con conjuntos de datos numéricos utilizando la distancia mahalanobis.

En la siguiente prueba se evalúa la calidad de los clústeres formados utilizando la distancia mahalanobis. La prueba consistió en variar el numero de clústeres a formar (K). Primero se muestra el comportamiento de la suma del promedio del error cuadrático de los atributos en cada clúster (SPECAC) en la tabla 60, esto con el fin de determinar el comportamiento de la SPECAC para observar su efecto en la dispersión dentro de los clústeres mostrando la grafica de comportamiento de la SPECAC con la variación de k en la figura 178, también se presentan en la tabla 61 los valores del promedio de la distancia euclidiana entre los centros de clústeres para determinar el comportamiento de la semejanza entre ellos, es decir que tan separados están los clústeres. Para ello también se realizo la figura 179, en donde se observa la tendencia de la SPECAC.

Tabla 60: SPECAC de los algoritmos variando el valor de k con MUNDODES utilizando la distancia mahalanobis.

Valor de k	Algoritmos		
	KMeans	K-Prototype	K-Frequency
K=2	74282402,634	74282402,634	74282402,634
K=4	155105190,922	115256238,176	161314276,235
K=5	221936344,599	161826752,792	218874717,100
K=8	207436354,640	228787834,983	258327190,185
K=16	411283422,258	253676930,413	322910028,348

Figura 178: Error cuadrático de los algoritmos particionales, variando k utilizando la distancia mahalanobis, aplicados a MUNDODES.

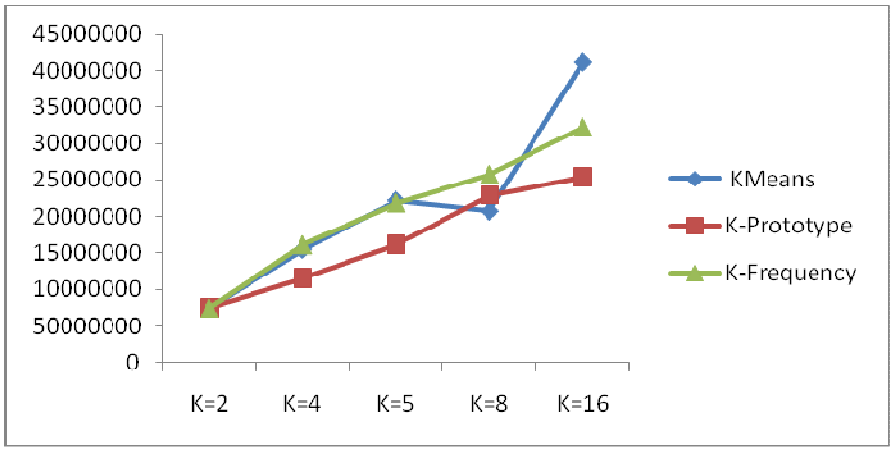
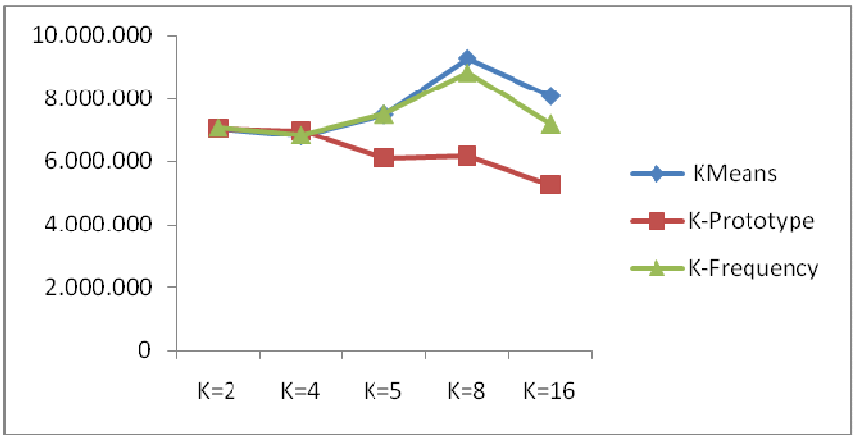


Tabla 61: Promedio de distancia entre los centros de clúster formados por los algoritmos particionales variando el valor de k, utilizando la distancia mahalanobis aplicados a MUNDODES.

Valor de k	Algoritmos		
	KMeans	K-Prototype	K-Frequency
K=2	7058.918	7058.918	7058.918
K=4	6806.666	6963.026	6827.667
K=5	7489.871	6119.886	7490.614
K=8	9251.735	6174.270	8822.260
K=16	8073.880	5268.797	7190.781

Figura 179: Promedio de distancia entre los centros de clúster formados por los algoritmos particionales, variando k utilizando la distancia mahalanobis, aplicados a MUNDODES.



Analizando los resultados de las pruebas antes mencionadas, se determina que al aumentar el número de clústeres, se aumenta la suma de promedio del error cuadrático de clúster SPECAC, presentado un comportamiento diferente al presentado con la utilización de la distancia euclidiana, eso debido que la distancia mahalanobis es considerada la distancia euclidiana estandarizada (ver capitulo 2.2.1). A diferencia de la figura 171, en la figura 178 no se encuentra un punto de baja reducción de la SPECAC, además los valores presentados por los algoritmos empiezan a ser diferentes en algunos valores de k.

Analizando los datos de la tabla 61, en donde se encuentran los resultados de la distancia promedio entre los centros de grupo formados al ir variando el valor de k, se puede establecer que la distancia entre ellos presenta un comportamiento irregular con respecto a el valor de k (ver figura 179). También se puede observar que los valores de la tabla 61 son inferiores en los primeros valores de k, con respecto a los datos de la tabla 57, aunque el valor de la distancia utilizando la distancia euclidiana termina siendo inferior al valor de la distancia mahalanobis.

8.2.4 Funcionalidad de los algoritmos de clustering particional con k=5 utilizando la distancia mahalanobis.

El objetivo de esta prueba es hacer análisis de las características de los representantes de cada clúster resultantes con los algoritmos kmeans, kprototype y kfrequency presentados en las tablas 62 ,63 y 64 respectivamente, utilizando la distancia mahalanobis y se lo hace con k=5 para comparar con los resultados del capítulo 8.2.2. Aquí todos los algoritmos de tipo particional obtuvieron resultados diferentes en la SPECAC.

Tabla 62: Representantes de cada clúster formados con k=5 en el algoritmo kmeans utilizando la distancia mahalanobis

Cluster	Tasa_Na	Tasa_Mort	Mort_Inf	Esp.Hom	Esp.Muj	PNB	Por.
Cluster 0	25,94	5,67	27,58	67,52	71,96	5242,22	19,78%
Cluster 1	13,42	11,19	12,13	69,15	76,31	11697,29	15,38%
Cluster 2	13,78	8,64	10,64	70,73	77,58	14224,27	16,48%
Cluster 3	45,59	17,73	115,98	47,46	50,34	443,71	23,08%
Cluster 4	37,48	9,4	76,92	58,47	61,93	1810,87	25,27%

Tabla 63: Representantes de cada clúster formados con k=5 en el algoritmo kprototype utilizando la distancia mahalanobis.

Cluster	Tasa_Na	Tasa_Mort	Mort_Inf	Esp.Hom	Esp.Muj	PNB	Por.
Cluster 0	23,66	5,8	22,66	68,7	73,44	7814	21,98%
Cluster 1	44,91	21,26	133,18	43,96	46,98	476,44	9,89%
Cluster 2	45,83	14,74	104,43	50,59	53,4	460,71	15,38%
Cluster 3	13,67	10,18	11,91	69,49	76,57	12085,62	28,57%
Cluster 4	36,66	9,01	73,05	59,14	62,67	1873,18	24,18%

Tabla 64: Representantes de cada clúster formados con k=5 en el algoritmo kfrequency utilizando la distancia mahalanobis.

Cluster	Tasa_Na	Tasa_Mort	Mort_Inf	Esp.Hom	Esp.Muj	PNB	Por.
Cluster 0	26,52	5,95	31,45	66,69	71,19	4671,43	23,08%
Cluster 1	13,42	11,19	12,13	69,15	76,31	11697,29	15,38%
Cluster 2	13,78	8,64	10,64	70,73	77,58	14224,27	16,48%
Cluster 3	45,53	17,99	115,88	47,68	50,38	425,4	21,98%
Cluster 4	38,99	9,8	82,05	57,26	60,68	1843,81	23,08%

En la tablas 65, 66, 67 se presentan los valores de la SPECAC dados por los algoritmos, para hacer un análisis de incidencia de los atributos en el resultado.

Tabla 65: SPECA aplicando el algoritmo kmeans con k = 5 y utilizando la distancia mahalanobis.

Cluster	Tasa_Na	Tasa_Mort	Mort_Inf	Esp.Hom	Esp.Muj	PNB	Total Sum
Cluster 0	40,13	1,92	232,5	10,24	13,33	37351683,95	37351982,07
Cluster 1	2,32	0,77	38,72	8,33	4,49	98374075,78	98374130,41
Cluster 2	9,33	0,81	32	10,72	8,23	82666590,86	82666651,95
Cluster 3	26,09	12,29	895,4	33,45	32,91	267133,06	268133,21
Cluster 4	43,13	2,57	479,51	11	15,85	3274894,9	3275446,96
Total	121,01	18,36	1678,13	73,75	74,81	221934378,55	221936344,6

Tabla 66: SPECA aplicando el algoritmo kprototype con k = 5 y utilizando la distancia mahalanobis en MUNDODES

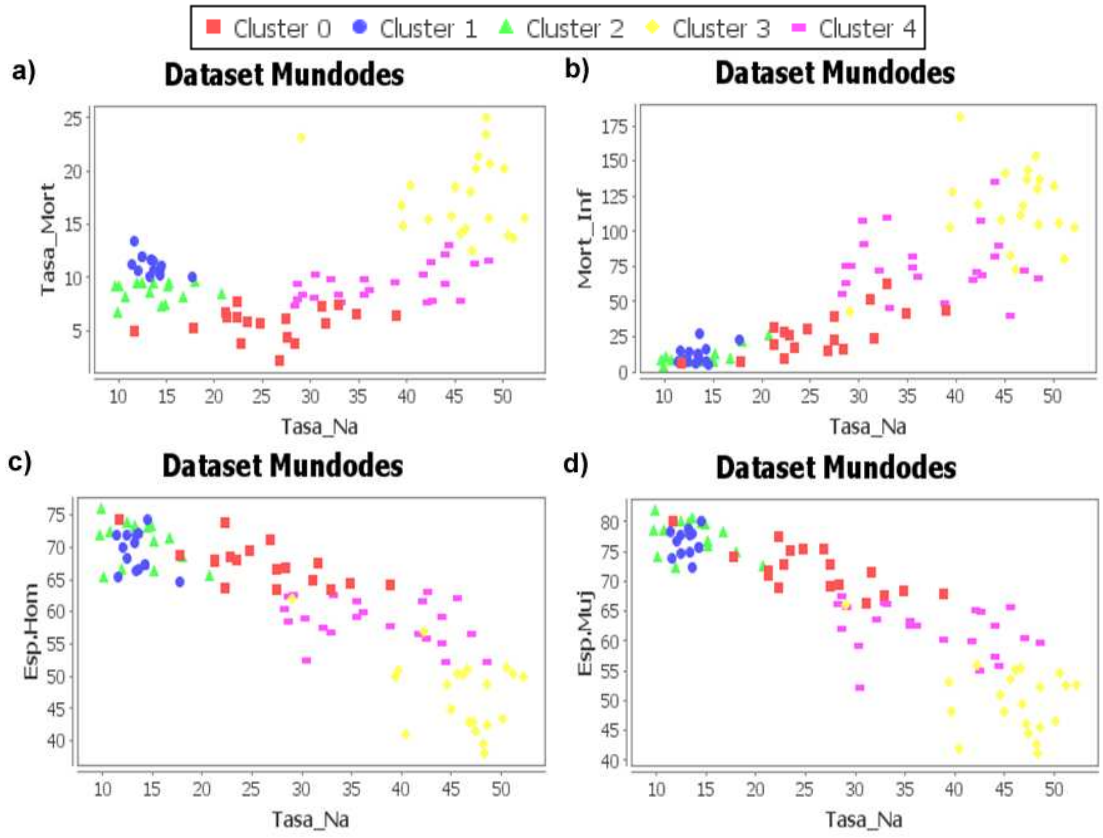
Cluster	Tasa_Na	Tasa_Mort	Mort_Inf	Esp.Hom	Esp.Muj	PNB	Total Sum
Cluster 0	54,02	1,9	188,89	13,65	19,69	64977994	64978272,14
Cluster 1	38,64	4,34	1234,3	44,92	49,88	528647,8	530019,88
Cluster 2	14,23	2,48	300,1	9,3	6,47	84635,2	84967,78
Cluster 3	6,04	1,71	36,71	9,01	5,98	92901264,62	92901324,08
Cluster 4	41,26	1,76	334,03	9,83	14,88	3331767,15	3332168,91
Total	154,19	12,19	2094,03	86,7	96,89	161824308,78	161826752,79

Tabla 67: SPECA aplicando el algoritmo kfrequency con k=5 y utilizando la distancia mahalanobis en MUNDODES

Cluster	Tasa_Na	Tasa_Mort	Mort_Inf	Esp.Hom	Esp.Muj	PNB	Total Sum
Cluster 0	37,17	2,13	296,95	13	15,01	34121117,01	34121481,26
Cluster 1	2,32	0,77	38,72	8,33	4,49	98374075,78	98374130,41
Cluster 2	9,33	0,81	32	10,72	8,23	82666590,86	82666651,95
Cluster 3	27,32	11,46	939,95	34,03	34,52	273446,04	274493,34
Cluster 4	40,47	2,66	500,61	20,49	19,95	3437375,96	3437960,14
Total	116,61	17,83	1808,23	86,58	82,2	218872605,65	218874717,1

Para tener una mejor visión de lo anteriormente del resultado de la distancia mahalanobis, se presenta en la figura 180 a la 186 los diagramas de dispersión de los atributos identificando los clústeres formados por el algoritmo kmeans.

Figura 180: Graficas de dispersión mostrando los clústeres formados por kmeans con k=5 utilizando la distancia mahalanobis del atributo Tasa_Na con respecto a los atributos: a) Tasa_Mort, b) Mort_Inf, c) Esp.Hom, d) Esp.Muj, e) PNB.



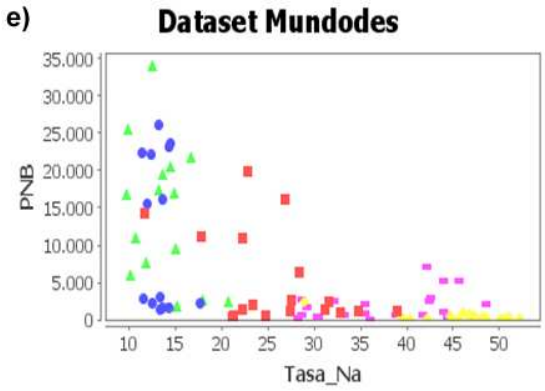


Figura 181: Graficas de dispersión mostrando los clústeres formados por kmeans con k=5 utilizando la distancia mahalanobis del atributo Tasa_Mort con respecto a los atributos: a) Mort_Inf, b) Esp.Hom, c) Esp.Muj, d) PNB.

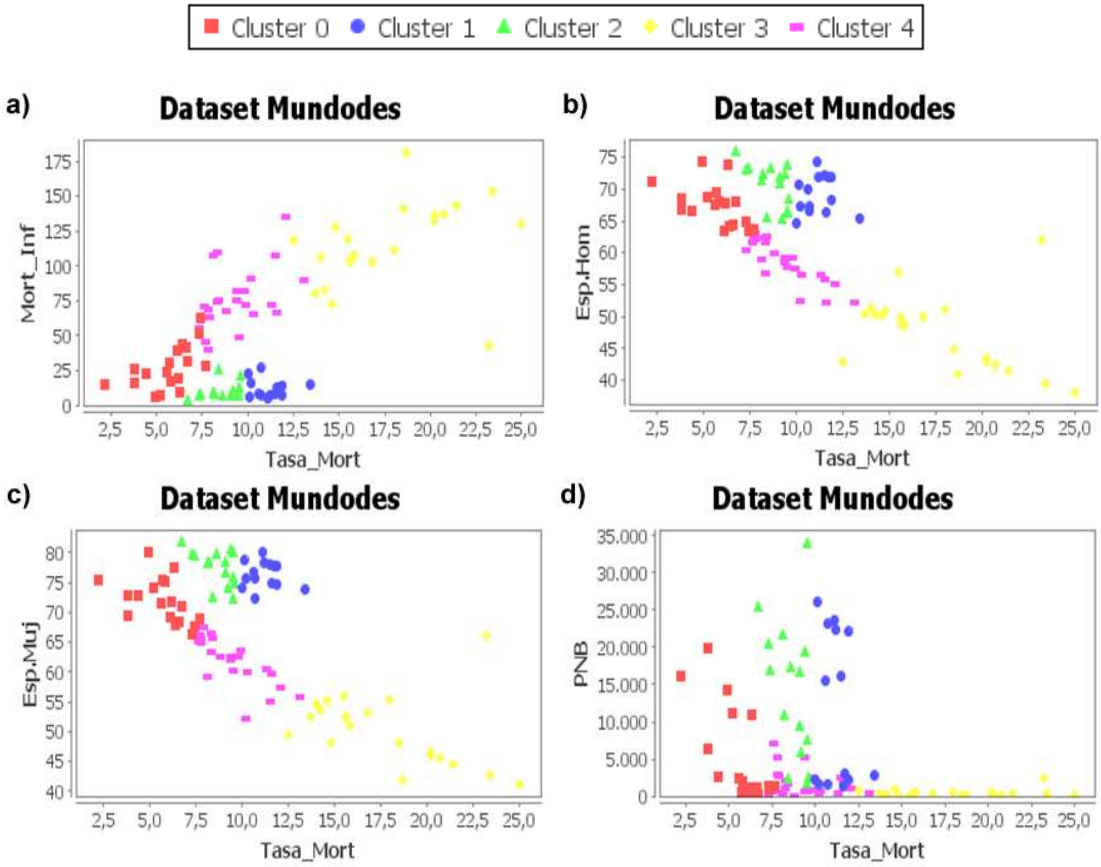


Figura 182: Graficas de dispersión mostrando los clústeres formados por kmeans con k=5 utilizando la distancia mahalanobis del atributo Mort_Inf con respecto a los atributos: a) Esp.Hom, b) Esp.Muj, c) PNB.

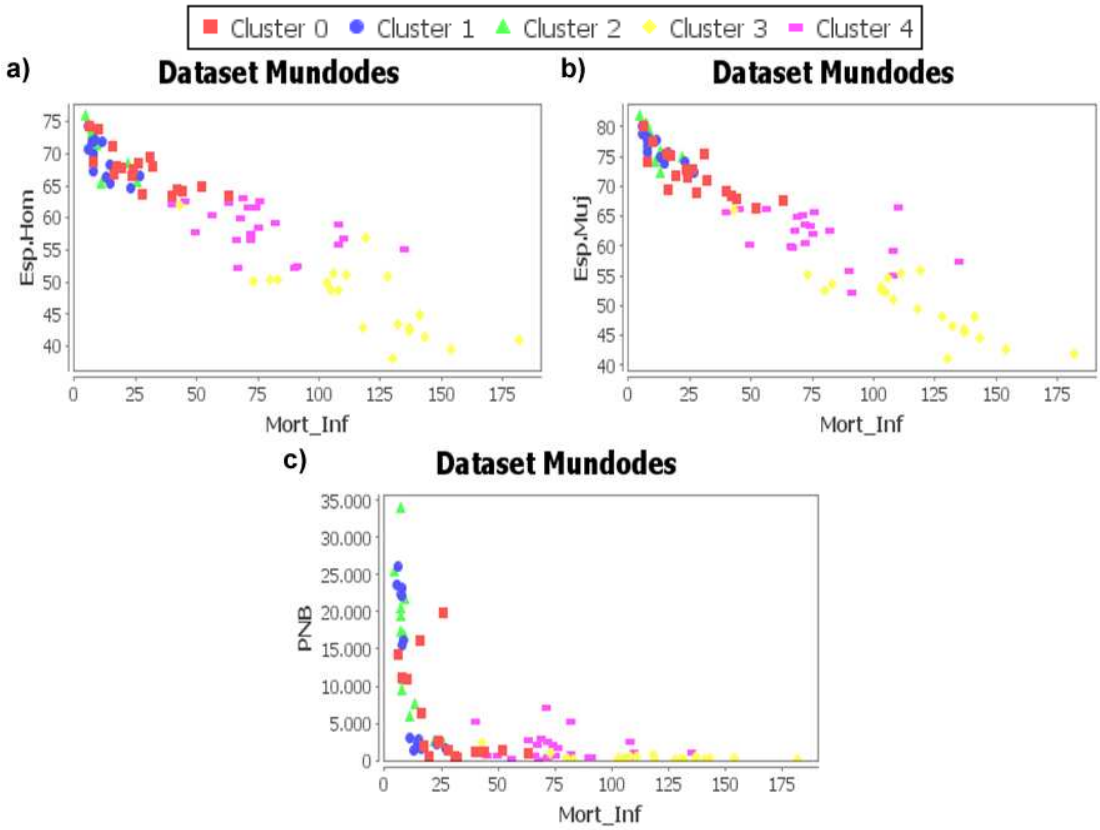


Figura 183: Graficas de dispersión mostrando los clústeres formados por kmeans con k=5 utilizando la distancia mahalanobis del atributo Esp.Hom con respecto a los atributos: a) Esp.Muj, b) PNB.

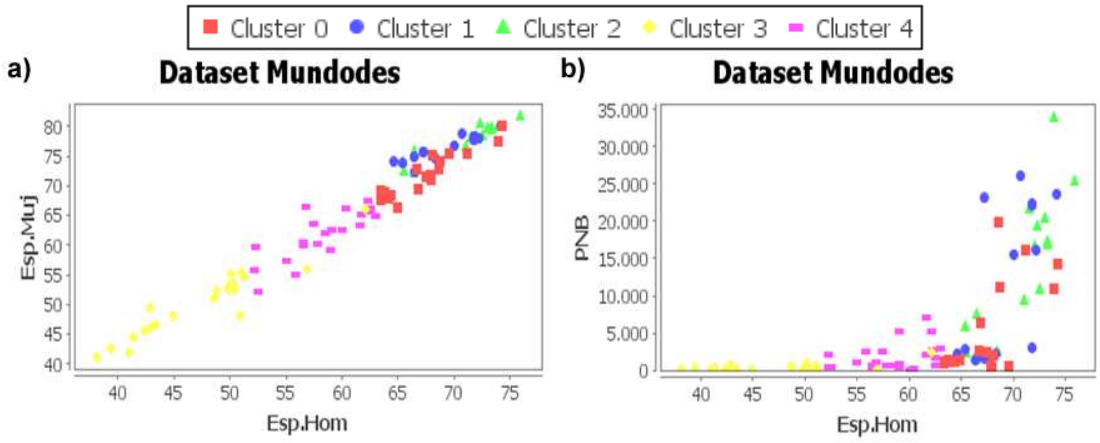
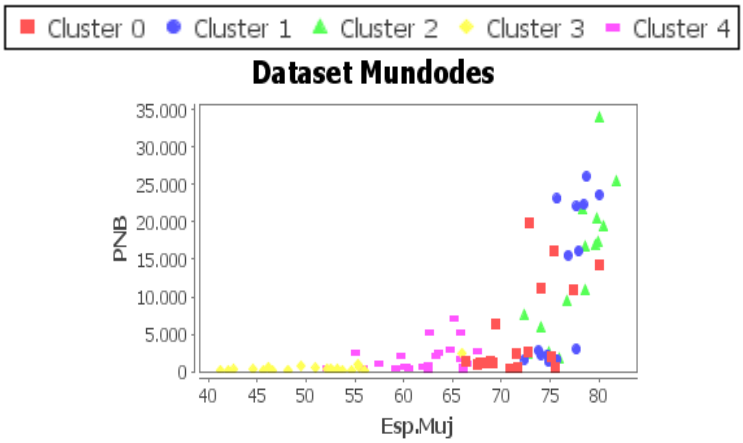


Figura 184: Graficas de dispersión mostrando los clústeres formados por kmeans con k=5 utilizando la distancia mahalanobis del atributo Esp.Hom con respecto a el atributo PNB.



Analizando los representantes de los clústeres de los algoritmos se puede destacar que los clústeres 1 y 2 que están 32% de los países estudiados, formados por el algoritmo kmeans y el algoritmo kfrequency, tienen un producto interno bruto per cápita superior a 11500. También se puede deducir que los países que tienen un bajo producto interno bruto es alta, la mortalidad infantil es baja y la esperanza de vida tanto de hombre como mujeres es alta.

Analizando más a fondo el comportamiento de la suma del promedio del error cuadrático de los atributos de cada clúster SPECAC con la distancia mahalanobis en el kmeans presentados en la tabla 65, los valores totales de error son inferiores en todos los atributos, lo que significa que utilizando la distancia mahalanobis, se tienen clústeres con la mayoría de sus atributos mas similares, creando así. Resultados más acordes con la similitud entre sus elementos, además observando las figuras 180,181, 182, 183 y 184 se puede ver que en la mayoría de los diagramas de dispersión de los atributos, están mejor definidos los clústeres producidos por la distancia mahalanobis, que los producidos por la distancia euclidiana presentados en la figuras 173, 174,175 y 176.

8.2.5 Calidad de clúster formados con los algoritmos de clustering particional con conjuntos de datos numéricos normalizados.

En la siguiente prueba se evalúa la calidad de los clústeres formados utilizando la distancia euclidiana, pero utilizando los datos aplicando el filtro que permite la normalización de los datos, con el objetivo de establecer las ventajas de la normalización. La prueba al igual que las anteriores consistió en variar el numero de clústeres a formar (K). Primero en la tabla 68 se muestra las características del

conjunto de datos resultante del filtro *normalization*, utilizando la desviación estándar de cada atributo, después se muestra la tabla 69 que contiene la suma del promedio del error cuadrático de los atributos en cada clúster (SPECAC) utilizando los algoritmos de tipo particional con los datos normalizados, enseguida se presenta en la tabla 70 el comportamiento de la SPECAC pero utilizando la asignación a los clústeres, para calcular la SPECAC con los datos, esto con el fin de determinar el comportamiento de los datos reales.

Tabla 68: Características de los atributos del conjunto de datos MUNDODES con atributos numéricos normalizados.

	Tasa_Na	Tasa_Mort	Mort_Inf	Esp.Hom	Esp.Muj	PNB
Mínimo	-1,442	-1,822	-1,097	-2,393	-2,231	-0,700
Media	0,000	-0,000	0,000	0,000	-0,000	-0,000
Máximo	1,660	3,045	2,728	1,493	1,417	3,499
Desviación Media	0,880	0,757	0,866	0,827	0,841	0,792
Desviación Estándar	1,000	1,0	1,0	1,000	1,000	1,000
Varianza	1,000	1,000	1,0	1,000	1,000	1,000
Suma de Error Cuadrático	90,000	90,000	90,0	90,000	90,000	90,000

Tabla 69: SPECA de los algoritmos variando el valor de k utilizando la distancia euclidiana, aplicados a MUNDODES normalizado utilizando los datos normalizados.

Valor de k	**Algoritmos**		
	KMeans	**K-Prototype**	**K-Frequency**
K=2	5,045	5,045	5,045
K=4	4,208	4,016	4,208
K=5	4,914	3,789	4,212
K=8	3,353	3,867	4,04
K=16	3,72	3,851	3,617

Tabla 70: SPECA de los algoritmos variando el valor de k utilizando la distancia euclidiana, aplicados a MUNDODES normalizado utilizando los datos originales.

Valor de k	**Algoritmos**		
	KMeans	**K-Prototype**	**K-Frequency**
K=2	84832899,733	84832899,733	84832899,733
K=4	39851985	37975725,043	39851985
K=5	41699150,095	40471356,667	49286263,876
K=8	43903139,516	44972537,665	44613687,647
K=16	48528809,119	47295215,139	45393692,680

Analizando el resultado de la normalización en la tabla 68, se observa que la suma del error cuadrático es aproximadamente de 90 y la desviación estándar es aproximadamente 1 en todos los atributos, lo que significa que todos tienen la misma dispersión e incidencia en el resultado del proceso de clustering, además se mira que la diferencia entre los valores entre los atributos el muy pequeña y se encuentran entre -2,393 que es el valor mínimo del atributo esp.Hom y 3,499 que es mayor valor perteneciente al atributo PNB**.**

Entrando en al análisis de la SPECAC de los datos normalizados de la tabla 69, se mira un comportamiento irregular de dichos datos en los resultados de los algoritmos y haciendo un contraste de la tabla 70 que contiene la SPECAC de los datos reales, con la tabla 56 que es la SPECAC resultado de los algoritmos particionales con los datos originales, se puede ver que los datos normalizados tienen un mayor valor de la SPECAC en todos los valores k. deduciendo que al tener en cuenta que el atributo PNB no tiene tanta incidencia en el resultado, su SPECAC puede ser más elevada e influir directamente el resultado total de este.

8.2.6 Funcionalidad de los algoritmos de clustering particional con k=5 utilizando MUNDODES normalizado.

El objetivo de esta prueba es hacer análisis de las características de los representantes de cada clúster resultantes con los algoritmos kmeans y kfrequency presentados en las tablas 71 y 72 respectivamente, utilizando la distancia euclidiana aplicándolos a el conjunto de datos MUNDODES normalizado y se lo hace con k=5 para comparar con los resultados del capítulo 8.2.2.

Tabla 71: Representantes de cada clúster formados con k=5 en el algoritmo kmeans utilizando la distancia euclidiana aplicado a MUNDODES normalizado.

Cluster	Tasa_Na	Tasa_Mort	Mort_Inf	Esp.Hom	Esp.Muj	PNB	Por.
Cluster 0	36,45	8,74	69,8	59,78	63,26	1800	27,47%
Cluster 1	15,4	13,24	19,91	66,81	73,59	2164,29	7,69%
Cluster 2	14,13	8,54	8,71	72,09	78,46	20131,26	20,88%
Cluster 3	46,22	17,02	118,98	47,35	50,2	374,91	24,18%
Cluster 4	20,92	7,16	20,17	67,34	73,17	3975,67	19,78%

Tabla 72: Representantes de cada clúster formados con k=5 en el algoritmo kfrequency utilizando la distancia euclidiana aplicado a MUNDODES normalizado.

Cluster	Tasa_Na	Tasa_Mort	Mort_Inf	Esp.Hom	Esp.Muj	PNB	Por.
Cluster 0	36,45	8,74	69,8	59,78	63,26	1800	27,47%
Cluster 1	13,9	10,53	15,5	67,32	74,75	3540,17	13,19%
Cluster 2	14,13	8,54	8,71	72,09	78,46	20131,26	20,88%
Cluster 3	45,47	17,29	115,68	47,99	50,89	466,87	25,27%
Cluster 4	24,05	5,99	22,79	67,48	72,43	3478,33	13,19%

En la tabla 73 y 74 se presenta la SPECAC generados por el algoritmo kmeans utilizando la distancia euclidiana, en la primera se muestra es el resultado de la SPECAC con los datos estandarizados y la segunda presenta el mismo resultado pero utilizando los datos originales de MUNDODES y la tabla 66 que es la unión de los clústeres pero utilizando para calcular la SPECA con los datos originales, con el fin de observar la incidencia de los atributos en la formación de clústeres y también para la este análisis se presentan los diagramas de dispersión de los atributos desde la figura 185 hasta la figura 189.

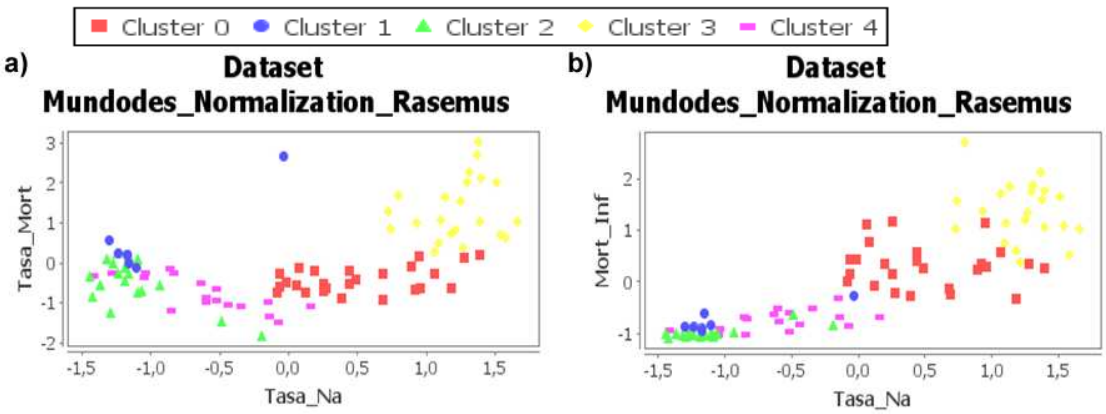
Tabla 73: SPECA en cada clústeres formados con k = 5 en el algoritmo kmeans utilizando la distancia euclidiana, aplicado a MUNDODES normalizado

Cluster	Tasa_Na	Tasa_Mort	Mort_Inf	Esp.Hom	Esp.Muj	PNB	Total Sum
Cluster 0	0,2	0,1	0,17	0,12	0,13	0,05	0,77
Cluster 1	0,17	0,79	0,05	0,08	0,1	0,01	1,19
Cluster 2	0,09	0,31	0,01	0,04	0,03	0,39	0,88
Cluster 3	0,06	0,56	0,3	0,27	0,19	0	1,39
Cluster 4	0,17	0,17	0,04	0,07	0,05	0,19	0,68
Total	0,69	1,93	0,58	0,58	0,5	0,64	4,91

Tabla 74: SPECA en cada clústeres formados con k = 5 en el algoritmo kmeans aplicado al MUNDODES normalizado, tomado los valores originales

Cluster	Tasa_Na	Tasa_Mort	Mort_Inf	Esp.Hom	Esp.Muj	PNB	Total Sum
Cluster 0	37,84	2,1	373,45	11,7	15,71	2973440	2973880,81
Cluster 1	31,47	17,4	110,81	7,39	11,87	338653,06	338831,99
Cluster 2	16,75	6,81	21,27	3,94	4,19	25805190,83	25805243,79
Cluster 3	12,18	12,28	650,72	25,78	23,94	75680,63	76405,52
Cluster 4	31,86	3,72	78,65	6,15	6,16	12504661,44	12504787,99
Total	130,1	42,31	1234,89	54,96	61,87	41697625,96	41699150,09

Figura 185: Graficas de dispersión mostrando los clústeres formados por kmeans con k=5 utilizando la distancia euclidiana del atributo Tasa_Na con respecto a los atributos: a) Tasa_Mort, b) Mort_Inf, c) Esp.Hom, d) Esp.Muj, e) PNB.



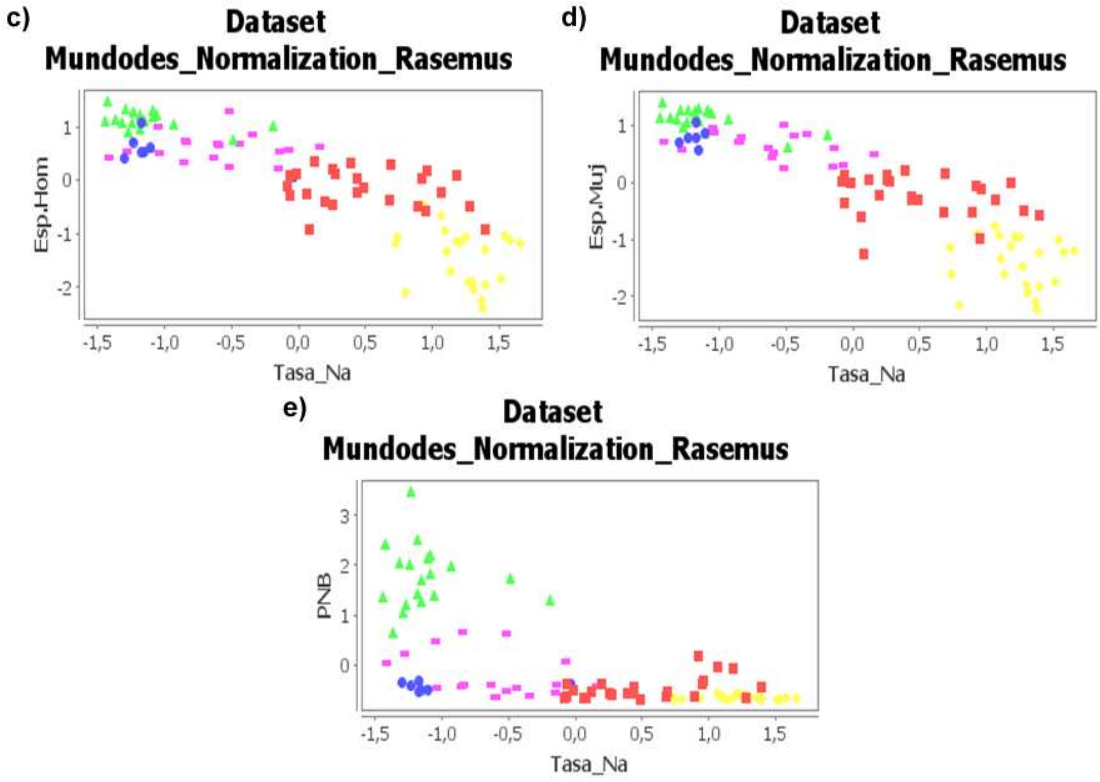
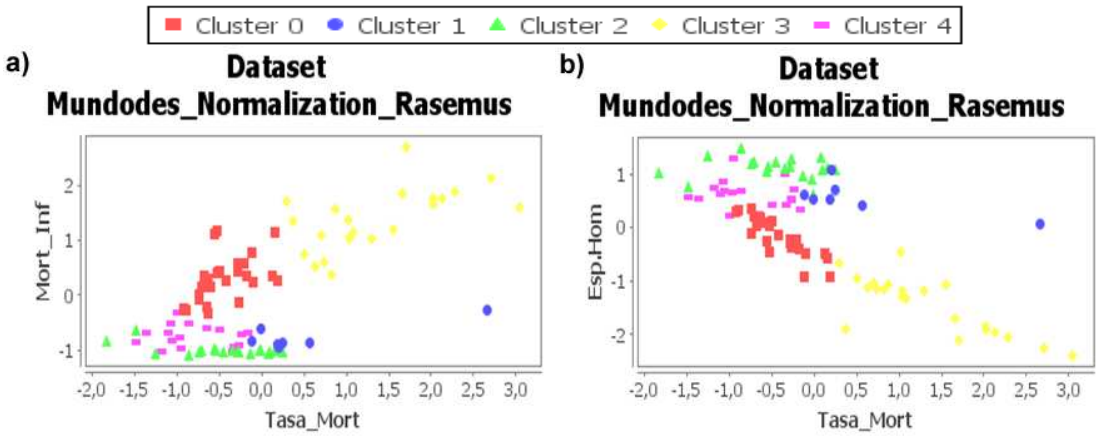


Figura 186: Graficas de dispersión mostrando los clústeres formados por kmeans con k=5 utilizando la distancia euclidiana del atributo Tasa_Mort con respecto a los atributos: a) Mort_Inf, b) Esp.Hom, c) Esp.Muj, d) PNB.



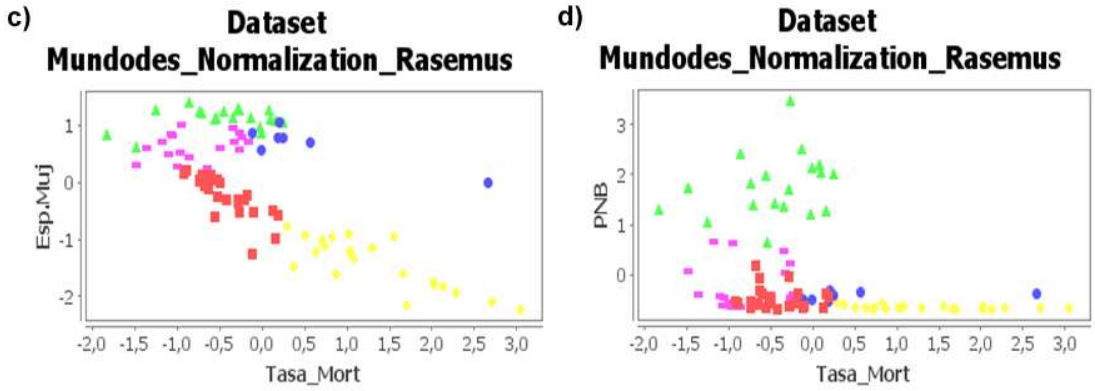


Figura 187: Graficas de dispersión mostrando los clústeres formados por kmeans con k=5 utilizando la distancia euclidiana del atributo Mort_Inf con respecto a los atributos: a) Esp.Hom, b) Esp.Muj, c) PNB.

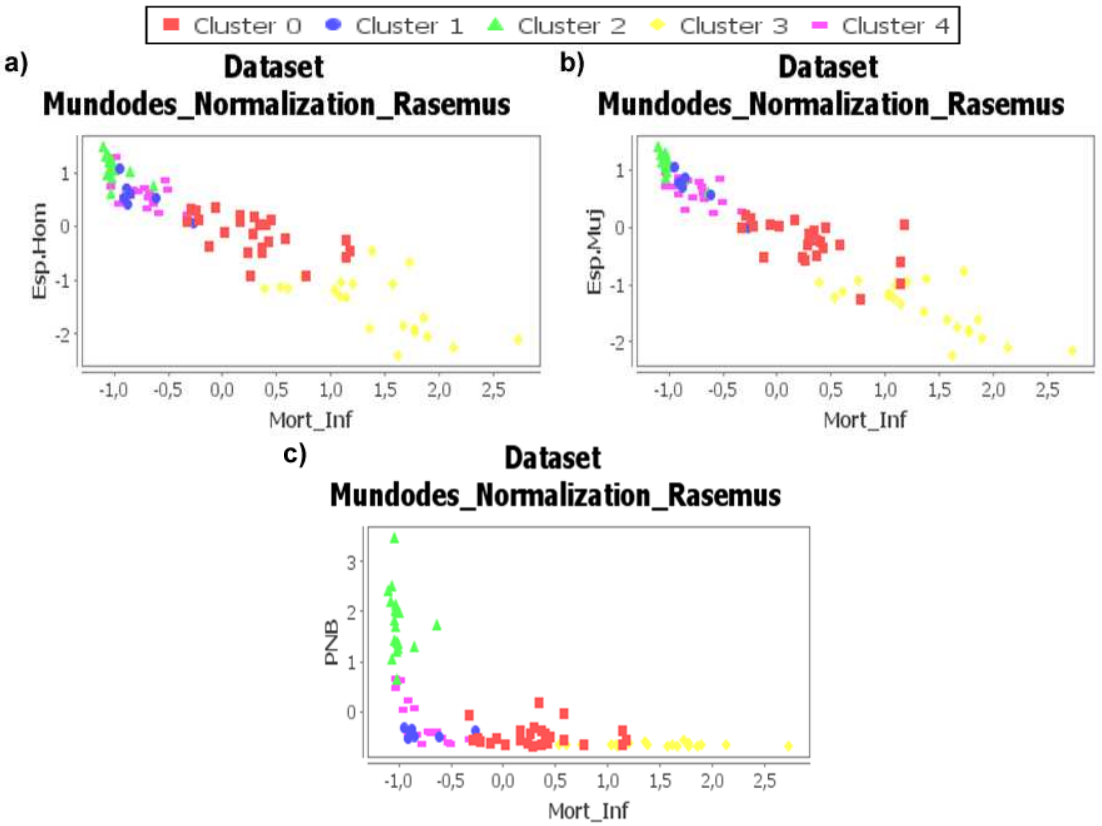


Figura 188: Graficas de dispersión mostrando los clústeres formados por kmeans con k=5 utilizando la distancia euclidiana del atributo Mort_Inf con respecto a los atributos: a) Esp.Muj, b) PNB.

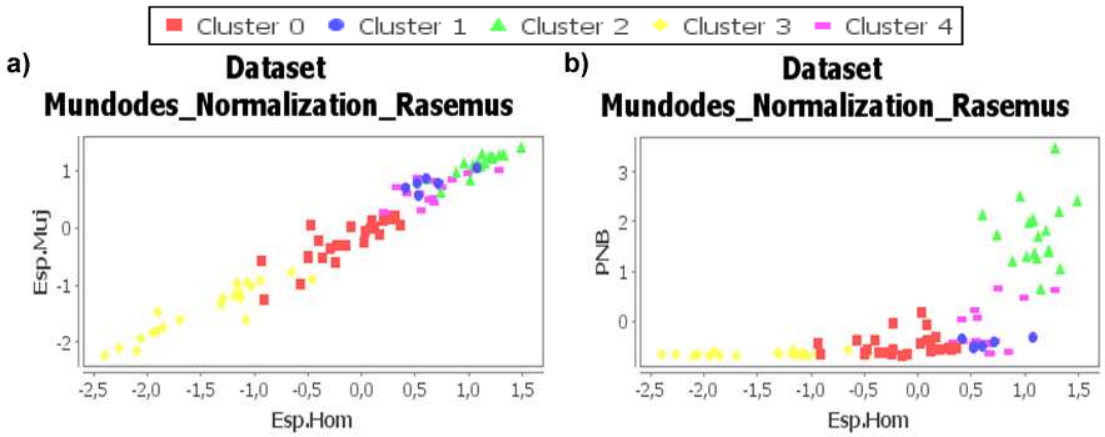
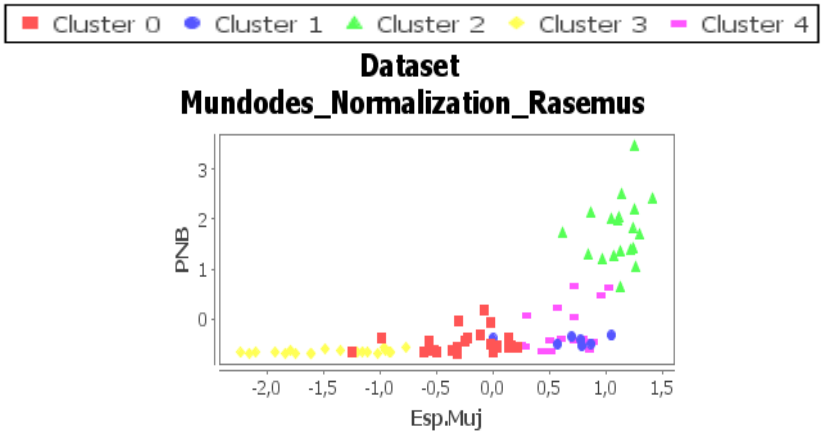


Figura 189: Graficas de dispersión mostrando los clústeres formados por kmeans con k=5 utilizando la distancia euclidiana del atributo Mort_Inf con respecto a el atributo PNB.



Analizando los representantes de clústeres formados por los algoritmos kmeans y kfrequency que se muestran en la tabla 72 y 73, se puede establecer que en el clúster 0 formado por kmeans, se ubican los países que tienen una tasa de natalidad baja, que su tasa de mortalidad y mortalidad infantil en inferior a 9.00, la expectativa de vida es superior a 70 años en hombres y mujeres y tienen un producto nacional bruto promedio de 20131.26 per cápita y que solo pertenecen a este clúster el 20.88% de los países estudiados. Este clúster también se presenta entre los clústeres formados por el algoritmo kfrequency, al igual que el cluster 0. También en los se encuentra entre los resultados del algoritmo kmeans el clúster 3 que presenta el valor del producto nacional bruto más bajo de todos los clústeres

formados hasta ahora, mostrando un valor medio de la tasa de mortalidad infantil de 118,98, pero que al igual se tiene una media de natalidad de 46,22 que es la más alta de todos los clústeres formados y presentado una media inferior a 51 de expectativa de vida tanto en hombres como en mujeres. Analizando la suma del promedio del error cuadrático de los atributos de cada clúster SPECAC del algoritmo kmeans, teniendo en la tabla 73 el valor de la SPECAC con los datos normalizados y la tabla 74 con la SPECAC de los datos originales, se observa que el atributo que más influye cuando se tiene los datos normalizados es el atributo Tasa_mort, pero que a su vez comparado con los valores los demás atributos no es muy alta. Al calcular la SPECAC con los datos originales se puede ver el mayor valor es el del atributo PNB, que incrementa significativamente el valor total de la SPECAC, pero revisando la SPECA de los demás atributos es considerablemente baja comparado con los resultados anteriores, creando así clústeres con una gran similitud entre sus miembros. Esto se puede ver en los diagramas de dispersión de la figuras 185, 186, 187, 188 y 189 en donde en la mayoría de diagramas se ve una mejor definición de los clústeres formados.

8.2.7 Funcionalidad de los algoritmos de clustering jerárquico datos numéricos y con k=5.

Al igual que la prueba anterior se varía el valor de k y se anotara el error cuadrático de sus resultados de los algoritmos de tipo jerárquico. Se varia el valor de clústeres a formar (k) y se revisara la SPECAC (ver tabla 75), además se ejecutara el algoritmo jerárquico de enlace mínimo con k=5, se presenta en los representantes de dicha clusterizacion en la tabla 76 y se presenta dendograma formado.

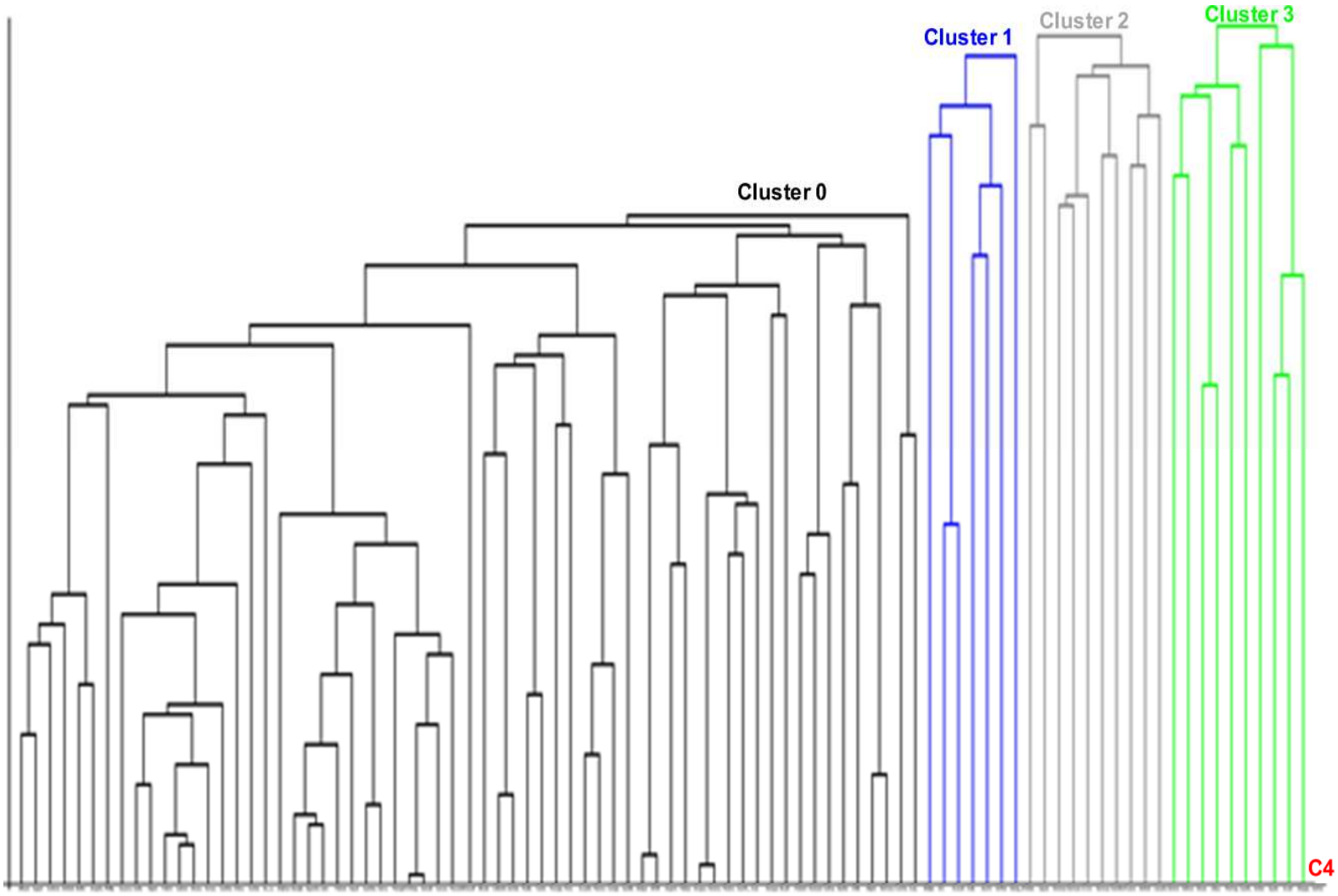
Tabla 75: SPECA de los algoritmos jerárquicos, variando el valor de k

Valor de k	Algoritmos		
	H-Single	H-Averange	K-Complete
K=2	56498068,972	29414458,481	29414458,481
K=4	18560831,983	18560831,983	12365192,949
K=5	11423682,253	11423682,253	11423682,253
K=8	4634577,927	3675839,291	3675839,291
K=16	1207487,674	847723,059	977063,792

Tabla 76: Representantes de algoritmos jerárquicos con k = 5.

Cluster	Tasa_Na	Tasa_Mort	Mort_Inf	Esp.Hom	Esp.Muj	PNB	Por.
Cluster 0	34.53	11.85	73.13	57.47	61.60	1124.13	69,23%
Cluster 1	14.56	7.76	8.84	72.33	78.33	16164.29	7,69%
Cluster 2	14.33	9.03	8.83	71.7	78.38	22426.0	10,99%
Cluster 3	24.8	7.61	26.59	66.78	71.61	8016.0	10,99%
Cluster 4	12.5	9.5	7.1	73.9	80.0	34064.0	1,10%

Figura 190: Dendograma realizado con el algoritmo jerárquico mínimo, determinando el valor de k=5.



También se presentan los diagramas de dispersión desde la figura 193 hasta la 198, con el fin de mirar el comportamiento en los atributos.

Figura 191: Graficas de dispersión resultado algoritmos de tipo jerárquico con k=5 utilizando la distancia euclidiana del atributo Tasa_Na con respecto a los atributos: a) Tasa_Mort, b) Mort_Inf, c) Esp.Hom, d) Esp.Muj, e) PNB.

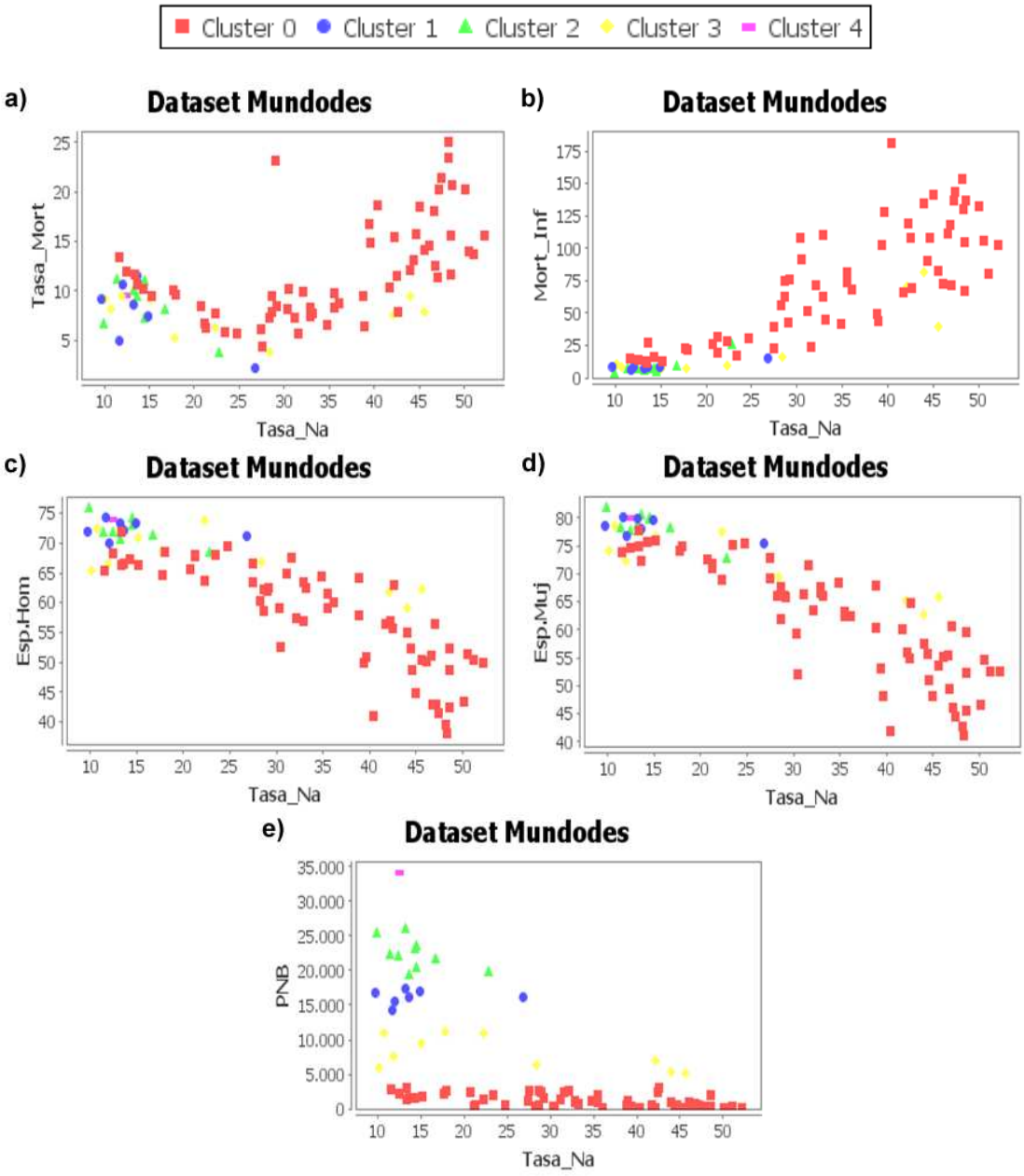


Figura 192: Graficas de dispersión resultado algoritmos de tipo jerárquico con k=5 utilizando la distancia euclidiana del atributo Tasa_Mort con respecto a los atributos: a) Mort_Inf, b) Esp.Hom, c) Esp.Muj, d) PNB.

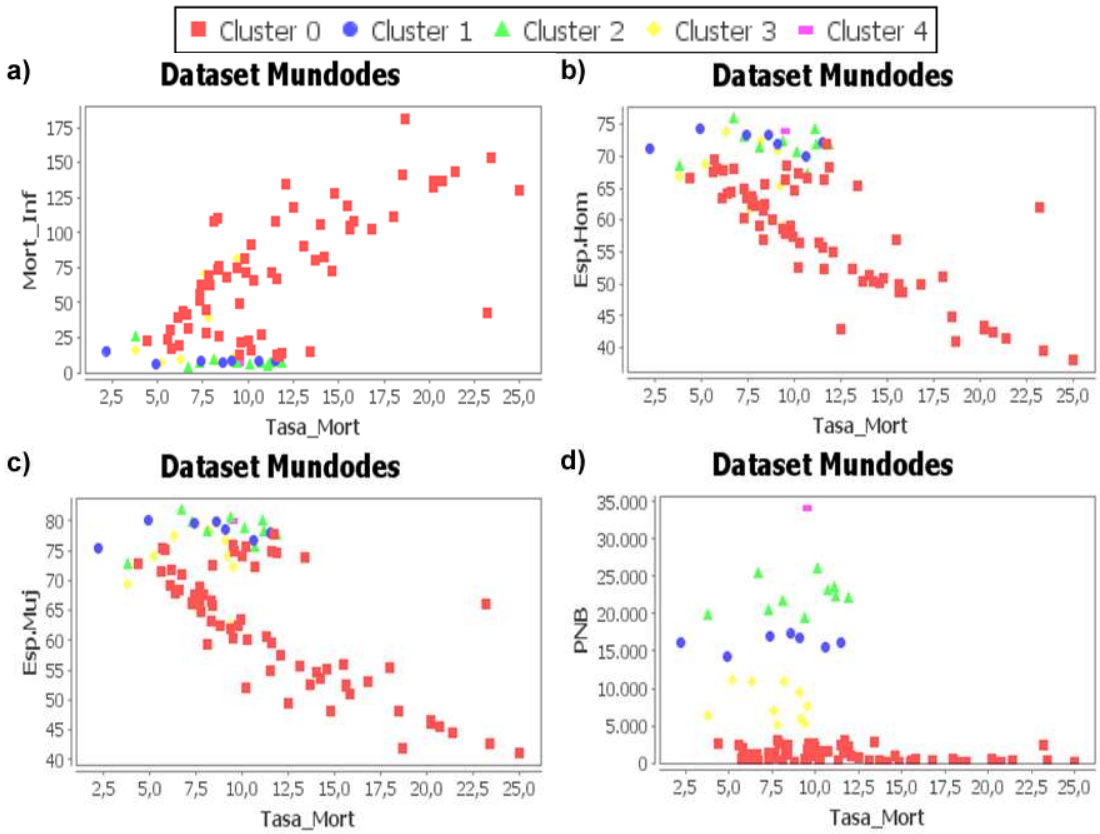
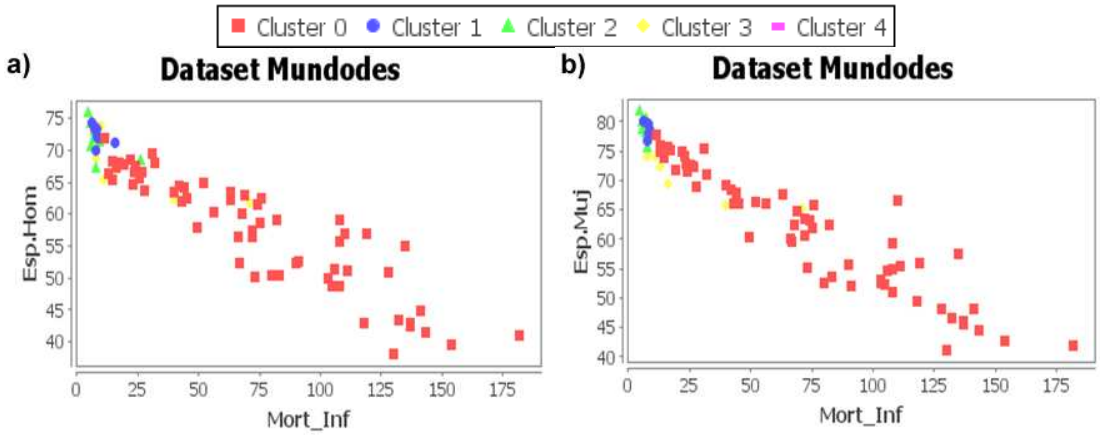


Figura 193: Graficas de dispersión resultado algoritmos de tipo jerárquico con k=5 utilizando la distancia euclidiana del atributo Mort_Inf con respecto a los atributos: a) Esp.Hom, b) Esp.Muj, c) PNB.



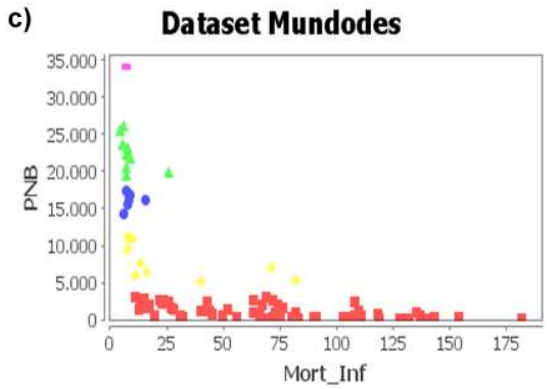


Figura 194: Graficas de dispersión resultado algoritmos de tipo jerárquico con k=5 utilizando la distancia euclidiana del atributo Esp.Hom con respecto a los atributos: a) Esp.Muj, b) PNB.

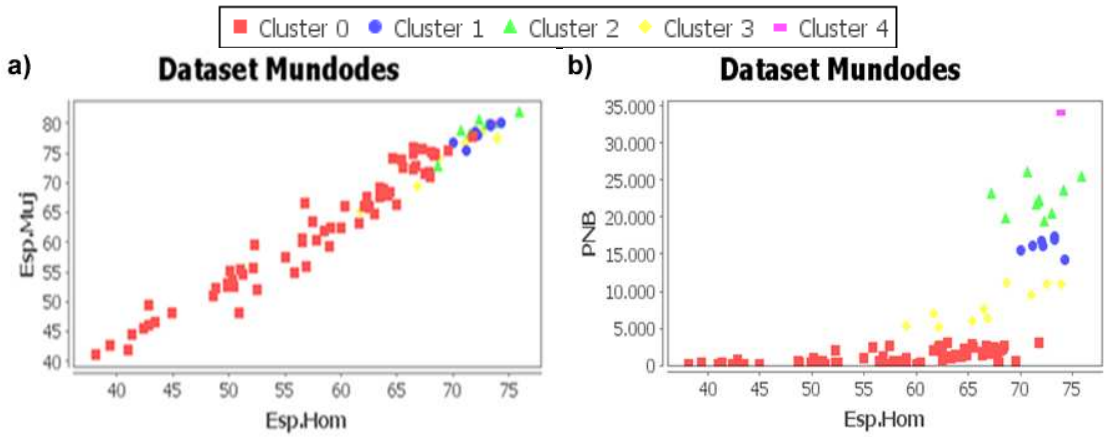
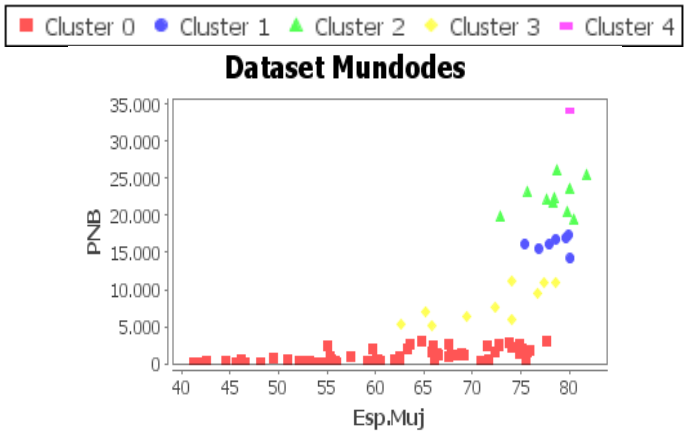


Figura 195: Graficas de dispersión resultado algoritmos de tipo jerárquico con k=5 utilizando la distancia euclidiana del atributo Esp.Muj con respecto a el atributo PNB.



Analizando los resultados de la tabla 75 se puede observar que el valor de la SPECAC siempre disminuye. En cuanto a los clústeres formados que se presenta

en la tabla 76, se puede decir que el clúster 0 es el que tiene aproximadamente un 70% de los países estudiados, piases que tienen un producto nacional bruto inferior a los 2000, y tienen una esperanza de vida tanto de hombre como de mujeres menor a 62 años. Por otra parte se tiene el clúster 4 que su porcentaje equivale a un registro, debido a que los valores de este registro son atípicos en todos sus atributos. El dendograma en la figura 190 muestra claramente las relaciones entre los clústeres y los registros que son más similares. Teniendo en cuenta las graficas de dispersión, se puede deducir que los algoritmos jerárquicos tienen un mejor manejo de datos anómalos.

8.3 PRUEBAS CON CONJUNTOS DE DATOS CON ATRIBUTOS CATEGÓRICOS.

Se realizaran dos tipos de pruebas utilizando dos repositorios uno de ellos es ZOO que es un repositorio sintético descargado de la UCI [106] y el otro es el conjunto de datos real denominado UDENAR, formado por el grupo de investigación GRIAS de la línea KDD en 2006 para mirar los patrones de bajo rendimiento y deserción de los estudiantes de la Universidad de Nariño (Colombia).

El conjunto de datos ZOO está formado por 17 atributos y 101 registros (ver tabla 77, En la prueba no se tendrá en cuenta el atributo animal ya que es el nombre que identifica al animal y solo se tomaran los registros que sean de tipo mamífero, con el fin de mirar la distribución de los animales son 41 registros,

Tabla 77: Atributos conjunto de datos Zoo

id	Atributo	Valores
--	Animal	Los nombre de los animales
Pe	Pelo	{no,si}
Pl	Plumas	{no,si}
Hu	Huevos	{no,si}
Le	Leche	{no,si}
Ae	Aéreo	{no,si}
Ac	Acuático	{no,si}
Dep	Depredador	{no,si}
Den	Dentadura	{no,si}
Col	ColumnaVerte	{no,si}
Res	Respiración	{no,si}
Ven	Venenoso	{no,si}
Ale	Aletas	{no,si}
Cola	Cola	{no,si}
Dom	Domestico	{no,si}
Peq	Pequeño	{no,si}
--	Tipo	{mamífero }

El conjunto de datos UDENAR consta de 20328 registros de estudiantes y 26 atributos, la mayoría de ellos está discretizados (ver anexo 1). La descripción de los atributos se presenta en la tabla 78.

Tabla 78: Atributos conjunto de datos UDENAR

id	Atributo	Valores	Descripción
E	edad	{A,B,C,D}	Es la edad actual del estudiante
Ei	edading	{A,B,C,D}	Es la edad de ingreso del es del estudiante
Fi	fecha_ing	{A,B,C,D,E}	Es la fecha de ingreso del estudiante
I	ingresos	{A,B,C,D,E}	El la cantidad de ingreso del núcleo familiar
vm	val_matricula	{A,B,C,D,E}	Valor de la matricula
Ne	nestrato	{0,1,2,3,4,5,6}	Estrato del recibo de luz
Na	naturaleza	{O,P}	Tipo de colegio
j	jornada	{C,M, T,N}	Jornada del programa que estudia
ca	calendario	{A,B,F}	En qué tipo d calendario ingreso
sx	sexo	{M,F}	Sexo o genero del estudiante
po	ponderado	{A,B,C,D}	Ponderado acumulado de notas
om	ocu_madre	{1,2,3,4,5}	Ocupación de la madre
mc	mas_una_a_cargo	{S,N}	tiene personas a cargo
vf	vive_con_familia	{S,N}	vive con su familia
hu	tiene_hermanos_u	{S,N}	hermanos en la Universidad
es	especial	{S,N}	Ingreso con cupo especial
pv	padre_vive	{S,N}	el padre vive
ec	estado_civil	{0,1,2,3,4}	estado civil de estudiante
mv	madre_vive	{S,N}	La madre vive
tr	tipo_residencia	{0,1,2,3}	La residencia donde vive es
cl	claseal	{1,2,3}	Determina que estudiantes han reingresado, se han retirado o no cumplen ninguna de las condiciones anteriores.
cr	claserend	{1,2,3,4,5}	Determina la cantidad de materias perdidas por el estudiante.
cp	clasepromedio	{A,B,C,D,E}	Determina el promedio acumulado del estudiante
zn	zona_nac	{N,NN, O,ON, O,ON, S,SN}	Zona de nacimiento
sm	semestred	{1,2,3,4,5}	Semestre al que está matriculado
cf	cod_facultad	{1,2,3,4,5,6,7,8,18,22,32}	Facultad a la cual pertenece

8.3.1 Funcionalidad de los algoritmos de clustering jerárquico con datos categóricos.

El objetivo de esta prueba es observar los resultados de los algoritmos jerárquicos cuando k=4, con el fin de mirar que clústeres se encuentran y analizar sus las características de sus representantes que se presentan en la tabla 79 los representantes de la agrupación jerárquica mínima, seguido de los representantes de la agrupación promedio en la tabla 80 y por último los representantes de la agrupación con enlace promedio en la tabla 81. Para ayudar al análisis se presentan los dendogramas resultantes en las figuras 196,197 y 198 respectivamente. Todos los algoritmos particionales utilizaron la distancia hammig.

Tabla 79: Representantes de cada clúster formados con k=u en el algoritmo jerárquico con enlace mínimo.

Cluster	Pe	Pl	Hu	Le	Ae	Ac	Dep	Den	Col	Res	Ven	Ale	Cola	Dom	Peq	%
Cluster 0	si	no	no	si	no	no	no	si	si	si	no	no	si	no	si	90,24%
Cluster 1	no	no	no	si	no	si	si	si	si	si	no	si	si	no	si	4,88%
Cluster 2	si	no	si	si	no	si	si	no	si	si	no	no	si	no	si	2,44%
Cluster 3	si	no	no	si	no	si	si	si	si	si	no	si	no	no	si	2,44%

Tabla 80: Representantes de cada clúster formados con k=4 en el algoritmo jerárquico con enlace promedio.

Cluster	Pe	Pl	Hu	Le	Ae	Ac	Dep	Den	Col	Res	Ven	Ale	Cola	Dom	Peq	%
Cluster 0	si	no	no	si	no	no	si	si	si	si	no	no	si	no	si	70,73%
Cluster 1	si	no	no	si	no	no	no	si	si	si	no	no	si	no	no	17,07%
Cluster 2	si	no	no	si	no	si	si	si	si	si	no	si	si	no	si	9,76%
Cluster 3	si	no	si	si	no	si	si	no	si	si	no	no	si	no	si	2,44%

Tabla 81: Representantes de cada clúster formados con k=4 en el algoritmo jerárquico con enlace máximo.

Cluster	Pe	Pl	Hu	Le	Ae	Ac	Dep	Den	Col	Res	Ven	Ale	Cola	Dom	Peq	%
Cluster 0	si	no	no	si	no	no	si	si	si	si	no	no	si	no	si	63,41%
Cluster 1	si	no	no	si	no	no	no	si	si	si	no	no	si	no	no	21,95%
Cluster 2	si	no	no	si	no	si	si	si	si	si	no	si	si	no	si	12,2%
Cluster 3	si	no	si	si	no	si	si	no	si	si	no	no	si	no	si	2,44%

Figura 196: Dendograma realizado con el algoritmo jerárquico mínimo, determinando el valor de k=4, aplicado a atributos de tipo categórico.

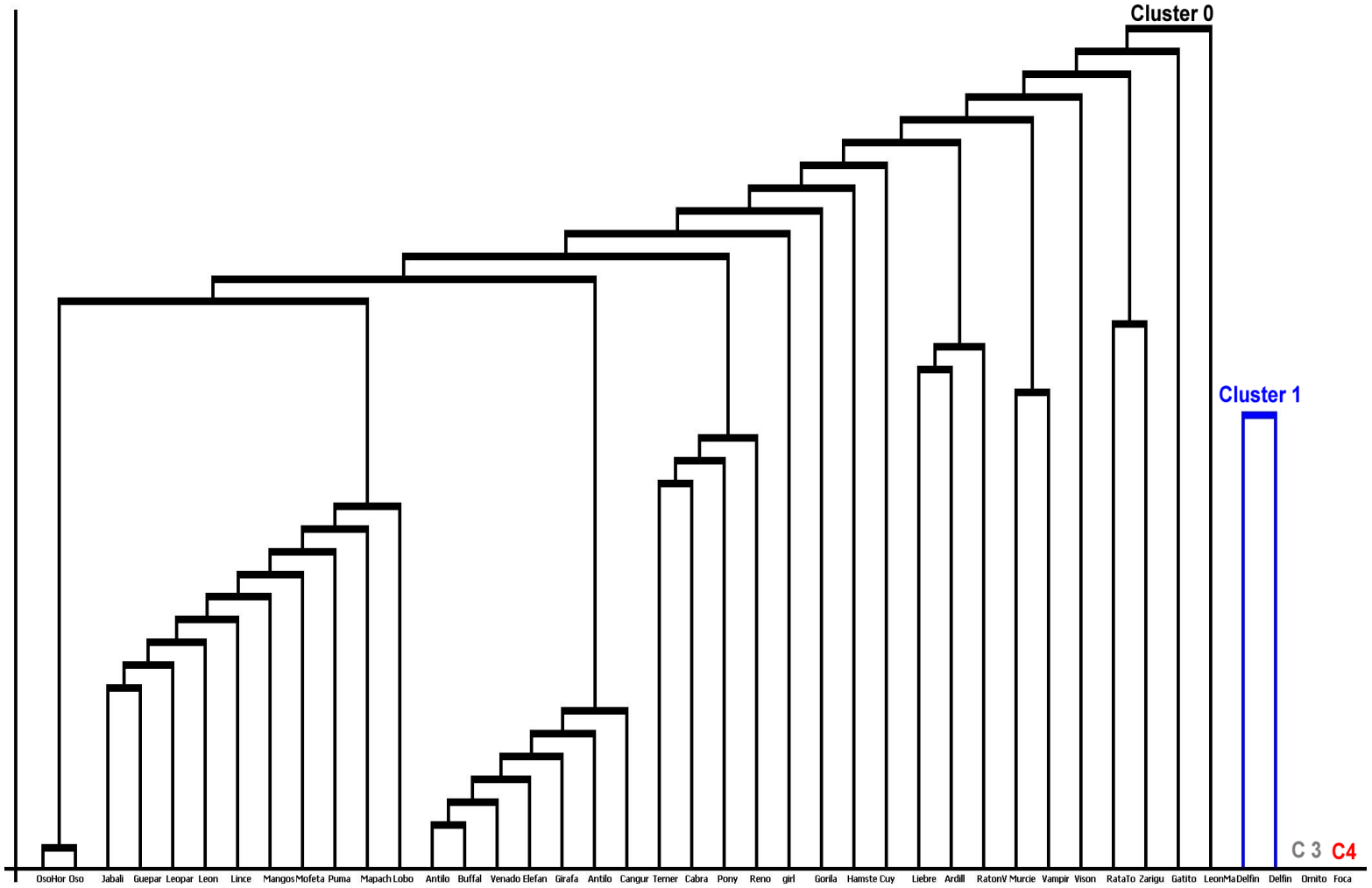


Figura 197: Dendograma realizado con el algoritmo jerárquico promedio, determinando el valor de k=4, aplicado a atributos de tipo categórico.

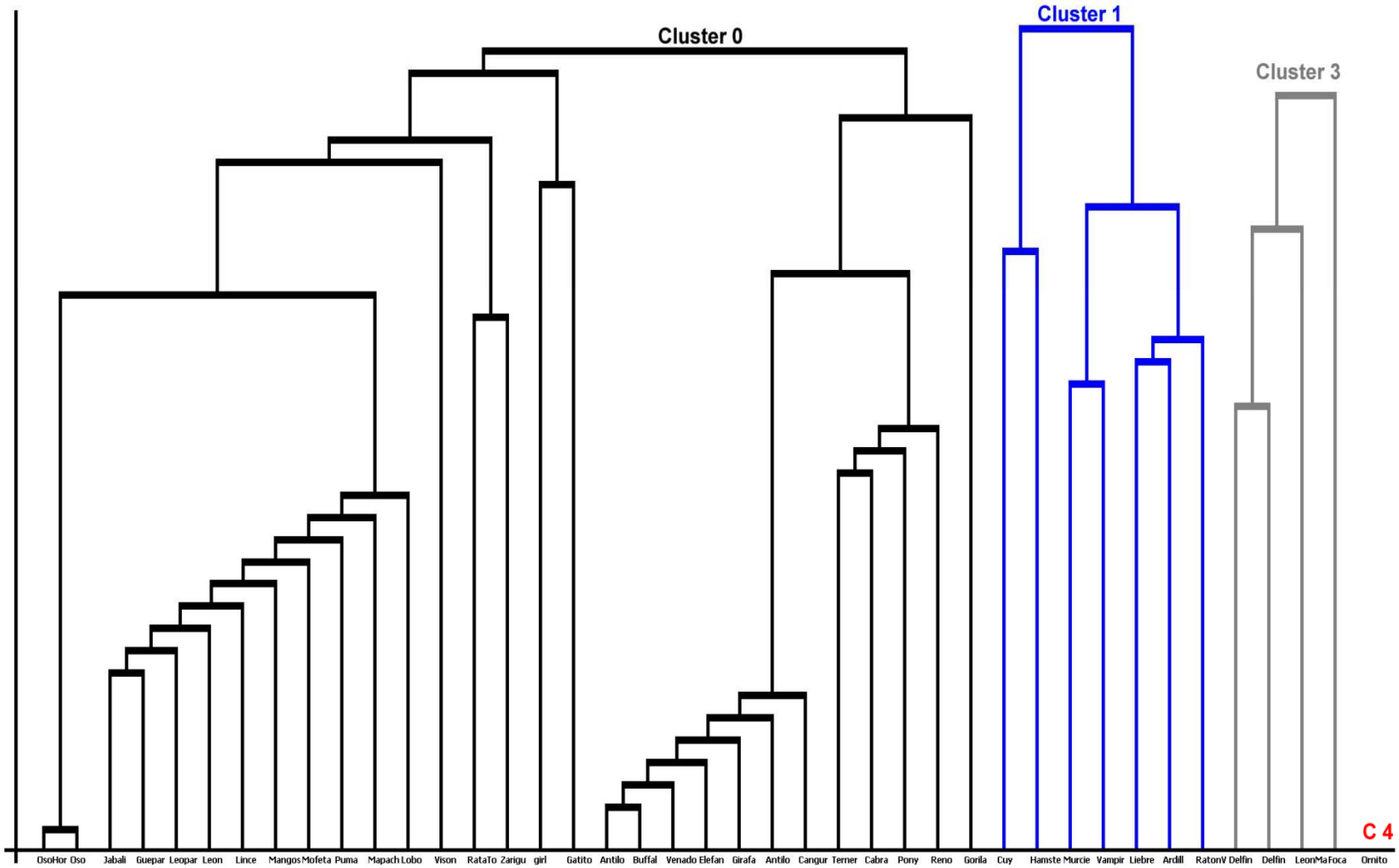
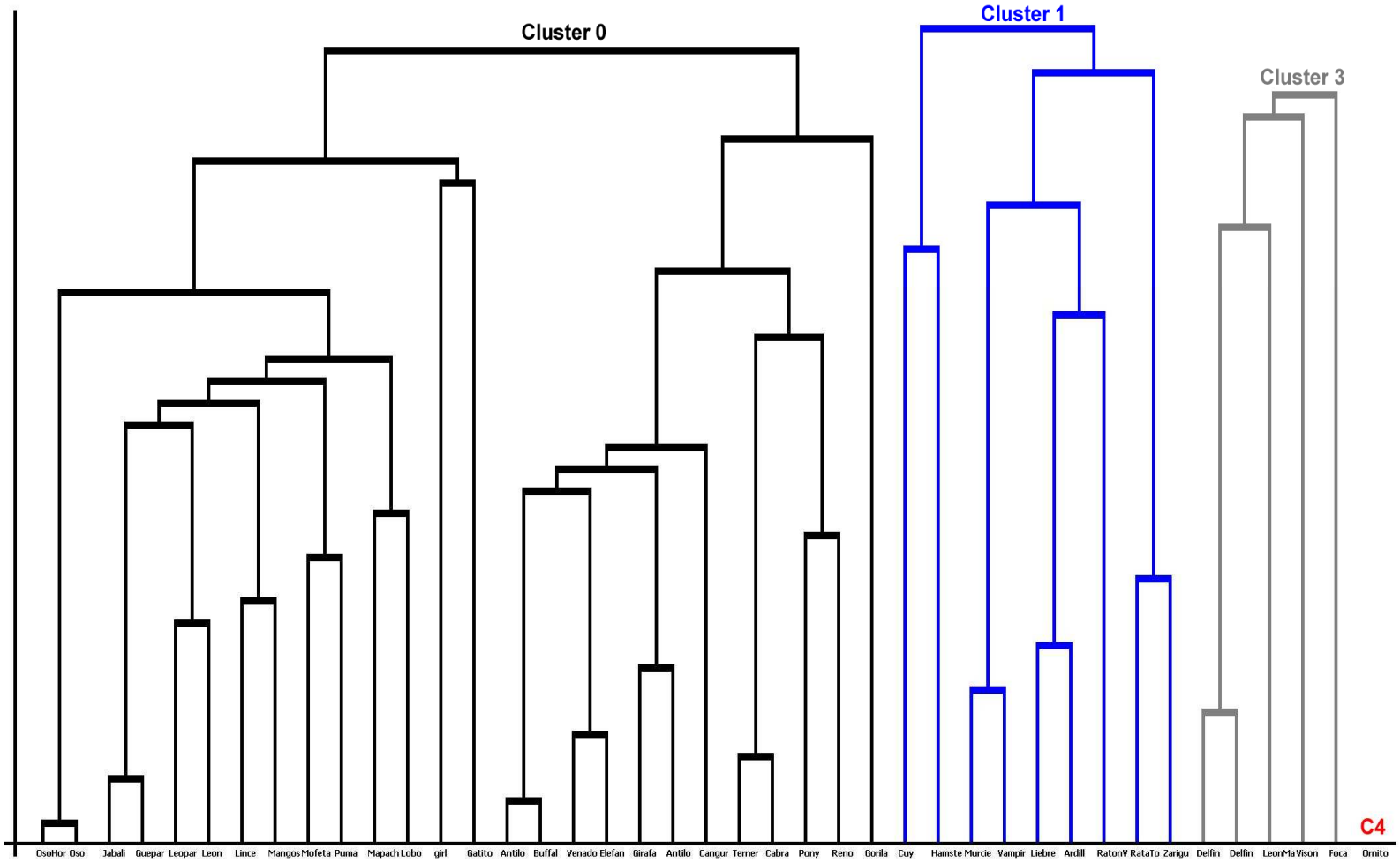


Figura 198: Dendograma realizado con el algoritmo jerárquico máximo, determinando el valor de k=4, aplicado a atributos de tipo categórico.



Analizando los resultados de los algoritmos jerárquicos se tiene que el animal que no tiene similitud con el resto de mamíferos es el ornitorrinco, que no se pudo agrupar con ninguno de los clúster, creando un clúster aparte. Entrando a analizar los resultados de cada algoritmo, el algoritmo de enlace mínimo, va haciendo uniones adicionando un elemento a la vez en los clústeres que se van formando (ver figura 196), creando asi un clúster con el 90% aproximadamente de los registros. En el caso del resultado con enlace promedio creo 4 clústeres con características especiales, el clúster 0 tiene todos los mamíferos grandes, el clúster 1 tiene todos los mamíferos que son pequeños, y la gran mayoría roedores y el clúster 3 tiene a todos los mamíferos de agua, como el delfín (ver figura 193). Para el caso del algoritmo con enlace máximo se tiene que el clúster 3 es similar al clúster 3 formado por el enlace promedio, en el clúster 1 incluyo a la zarigüeya, y en la formación del clúster 0 se pueden observar uniones entre los mamíferos mas similares creando subgrupos con un alto rango de similitud.

8.3.2 Funcionalidad de los algoritmos de clustering particional con datos categóricos con k = 6.

El objetivo es mirar que resultados arrojan los algoritmos de clustering partional con atributos categóricos con los algoritmos kmeans y kfrequency, para ello se utiliza el conjunto de datos UDENAR y se da un valor de k=6. La prueba se realizo utilizando la distancia hamming. En la tabla 82 y 83 se presentan los representantes de los clústeres formados por el algoritmo kmeans y kfrequency respectivamente.

Tabla 82: Representantes de cada clúster formados con k=6 en el algoritmo kmeans utilizando la distancia haming aplicado a UDENAR.

Cu	E	Ei	Fi	I	vm	Ne	Na	j	ca	sx	po	om	mc	vf	hu	es	pv	ec	mv	tr	cl	cr	cp	zn	sm	cf	%
C0	C	A	E	E	D	3	P	M	B	M	C	5	S	S	N	N	S	0	S	0	1	1	E	SN	1	8	10,4
C1	B	A	E	B	C	2	O	M	B	F	C	5	S	S	N	N	S	0	S	1	1	1	D	SN	1	8	28,52
C2	D	A	B	E	E	4	O	M	B	M	C	1	S	S	N	N	S	0	S	1	1	2	B	SN	1	7	9,15
C3	D	B	B	C	C	2	O	M	B	M	C	1	S	S	N	N	S	0	S	1	1	5	C	SN	5	7	25,45
C4	D	B	D	A	C	2	O	C	B	F	C	1	S	S	N	N	S	0	S	1	1	1	D	SN	1	8	18,13
C5	D	A	C	E	D	3	O	C	B	F	C	1	S	S	N	N	S	0	S	1	1	1	A	SN	1	8	8,34

Tabla 83: Representantes de cada clúster formados con k=6 en el algoritmo kfrequency utilizando la distancia haming aplicado a UDENAR

Cu	E	Ei	Fi	I	vm	Ne	Na	j	ca	sx	po	om	mc	vf	hu	es	pv	ec	mv	tr	cl	cr	cp	zn	sm	cf	%
C0	B	A	E	B	C	2	O	M	B	M	C	5	S	S	N	N	S	0	S	1	1	1	D	SN	1	8	20,93
C1	C	B	E	A	C	2	O	M	B	M	C	5	S	S	N	N	S	0	S	1	1	1	D	SN	2	8	9,15
C2	D	B	C	A	A	2	O	M	B	M	C	1	S	S	N	N	S	0	S	1	1	1	D	SN	5	8	13,87
C3	C	A	D	B	C	2	O	M	B	M	C	1	S	S	N	N	S	0	S	1	1	1	D	SN	1	8	20,73
C4	D	B	B	C	B	2	O	M	B	M	C	5	S	S	N	N	S	0	S	1	1	1	C	SN	5	7	24,01
C5	D	B	C	E	D	3	O	M	B	M	C	1	S	S	N	N	S	0	S	1	1	1	A	SN	1	7	11,33

En la tabla 84 y 85 se muestran los datos del valor de 1 – frecuencia del valor de la moda del clústeres en ese atributo, al que se denominara error de frecuencia del los atributos en cada cluster (EFAC) de los clústeres formados por los algoritmos, con el fin de establecer la calidad de los clústeres formados.

Tabla 84: EFAC en el algoritmo kmeans con k=6, aplicado a UDENAR.

Cu	E	Ei	Fi	I	vm	Ne	Na	j	ca	sx	po	om	mc	vf	hu	es	pv	ec	mv	tr	cl	cr	cp	zn	sm	cf	To
C 0	0,48	0,48	0,41	0,71	0,46	0,5	0,29	0,46	0,08	0,28	0,2	0,36	0,1	0,15	0,25	0,01	0	0,02	0	0,34	0,12	0,52	0,65	0,11	0,44	0,81	8,24
C 1	0,51	0,35	0,24	0,51	0,54	0,26	0,21	0,51	0,05	0,42	0,2	0,12	0,09	0,29	0,19	0,03	0	0,01	0	0,31	0,07	0,57	0,58	0,16	0,46	0,78	7,46
C 2	0,18	0,44	0,55	0,51	0,68	0,63	0,41	0,41	0,06	0,23	0,16	0,43	0,03	0,08	0,1	0,11	0,02	0,01	0	0,14	0,12	0,66	0,59	0,16	0,28	0,64	7,64
C 3	0,13	0,45	0,48	0,64	0,63	0,34	0,29	0,42	0,05	0,22	0,18	0,54	0,04	0,18	0,12	0,11	0,02	0,01	0,01	0,16	0,12	0,56	0,56	0,19	0,47	0,75	7,67
C 4	0,44	0,48	0,36	0,43	0,62	0,36	0,29	0,53	0,05	0,41	0,3	0,19	0,08	0,25	0,19	0,03	0,01	0,02	0	0,31	0,15	0,53	0,53	0,18	0,68	0,7	8,11
C 5	0,22	0,41	0,4	0,62	0,49	0,31	0,41	0,54	0,04	0,31	0,18	0,24	0,06	0,1	0,17	0,13	0,02	0,02	0,01	0,24	0,14	0,51	0,7	0,16	0,46	0,74	7,63
To	1,95	2,61	2,44	3,42	3,43	2,4	1,89	2,87	0,32	1,88	1,22	1,87	0,41	1,05	1,03	0,41	0,07	0,09	0,03	1,5	0,72	3,35	3,62	0,97	2,8	4,41	46,76

Tabla 85: EFAC en el algoritmo kfrequency con k=6, aplicado a UDENAR

Cu	E	Ei	Fi	I	vm	Ne	Na	j	ca	sx	po	om	mc	vf	hu	es	pv	ec	mv	tr	cl	cr	cp	zn	sm	cf	To
C 0	0,25	0,24	0	0,66	0,59	0,38	0,26	0,51	0,05	0,49	0,17	0	0,1	0,25	0,23	0	0	0,01	0	0,42	0,06	0,57	0,68	0,13	0,33	0,83	7,22
C 1	0,17	0,39	0,01	0,56	0,58	0,34	0,26	0,6	0,11	0,34	0,32	0	0,11	0,37	0,2	0	0	0,02	0	0,38	0,09	0,67	0,65	0,25	0,49	0,8	7,72
C 2	0,05	0,59	0,24	0,6	0,32	0,4	0,31	0,56	0,05	0,5	0,2	0,17	0,08	0,35	0,22	0,1	0,02	0,02	0	0,34	0,06	0,56	0,4	0,2	0,19	0,76	7,31
C 3	0,26	0,41	0,01	0,6	0,59	0,4	0,34	0,69	0,03	0,45	0,26	0	0,07	0,23	0,24	0	0	0,01	0	0,43	0,23	0,65	0,68	0,13	0,68	0,81	8,19
C 4	0	0,57	0,14	0,68	0,65	0,46	0,38	0,42	0,04	0,44	0,16	0,21	0,02	0,03	0,04	0,15	0,02	0,01	0,02	0,05	0,06	0,65	0,67	0,18	0,48	0,73	7,28
C 5	0,04	0,6	0,38	0,55	0,56	0,57	0,44	0,52	0,08	0,32	0,19	0,17	0,06	0,11	0,12	0,12	0,02	0,02	0	0,18	0,19	0,7	0,69	0,16	0,18	0,76	7,73
To	0,77	2,81	0,8	3,65	3,29	2,55	1,99	3,32	0,36	2,53	1,31	0,55	0,45	1,35	1,05	0,37	0,06	0,09	0,03	1,79	0,69	3,79	3,77	1,05	2,36	4,69	45,45

Analizando los resultados de las tabla 82 y 83, y utilizando el anexo 1 para reemplazar los valores discretizados por los valores originales y utilizando la frecuencia dentro de los clústeres, se tiene que el 28,52% de los estudiantes, se ubicaron en el clúster 1, en donde el aproximadamente el 49% tiene una edad mayor de 18 años y menor de 22, que el 65% ingreso con una edad menor o igual a 18 años, el 74% es de estrato 2, el 42% de ellos tiene un promedio mayor o igual a 3,5 hasta 4,0. Otro cluster que sobresale es el cluster 3 que hacen parte del 25,45% de los registros, y que tiene entre sus características las siguientes, que el 36% ingreso a la universidad después de1994 y antes del 2000, que han perdido más de 3 materias el 44% y el 53% esta en semestres inferiores al quinto.

En cuanto al algoritmo kfrequency se puede decir que los clústeres formados por este algoritmo, son diferentes de los formados por el algoritmo kmeans. Analizando la EFAC de los algoritmos kmeans y kfrecuency, en las tablas 84 y 85, se puede deducir que no hay superioridad del uno contra el otro, teniendo resultados similares.

8.4 PRUEBAS CON UN CONJUNTO DE DATOS CON ATRIBUTOS MIXTOS.

El objetivo de esta prueba es mirar el comportamiento de los algoritmos de tipo particional con conjuntos de datos de gran tamaño, y que contengan tipos de datos tanto categóricos como numéricos. Para ellos se utilizo el conjunto de datos SISBEN, que contiene información acerca de las condiciones de vida de las familias de una determinada región de Colombia. Este conjunto de datos consta de 100.000 registros y 30 variables de los cuales 6 de tipo numérico y 24 son de tipo categórico, la mayoría de ellos están discretizados. Para esta prueba solo se tomaran 21 atributos, los cuales se presentan en la tabla 86.

Tabla 86: Atributos conjunto de datos SISBEN21

id	Atributo	Valores	Descripción
z	zona	1,2,3	Que tipo de zona de asentamiento es.
v	vivienda	1,2,3	Tipo de unidad de vivienda
p	piso	1,2,3,4,5	Material predomínate en pisos
b	basura	1,2	Servicio de recolección de basura.
ac	acueducto	1,2	Servicio de acueducto.
e	estrato	0,1,2,3,4	Estrato del recibo de luz
tv	teneviv	1,2,3,4	La vivienda en la que viven es
tc	tcuartos	numérico	Cantidad de cuartos de la casa
td	tdormir	numérico	Cantidad de cuartos para dormir
s	sanitar	0,1,2,3,4	Servicio de sanitario que usan
cn	cocinan	0,1,2,3,4,5,6	Combustible con el que cocinan

tp	tpersonas	numeric	Número total de personas que conforman la familia
ec	estcivil	1,2,3,4,5	Estado civil
sx	sexo	1,2	Sexo o genero
as	asiste	1,2	Asiste a un centro educativo
te	tipoesta	0,1,2,3,4,5,6,7,8	Tipo de establecimiento educativo
ne	nivel	0,1,2,3,4,5	Nivel educativo
a	activi	0,1,2,3,4,5,6,7	Actividad en el último mes
ing	ingresos	numérico	Ingresos mensuales
ed	edad	numérico	edad
ns	NIVSIS	1,2,3,4	Nivel sisben asignado

8.4.1 Calidad de clúster formados con los algoritmos de clustering particional con conjuntos de datos mixtos.

El objetivo de esta prueba es mirar el comportamiento tanto de la suma promedio del error cuadrático de cada atributo en cada clúster (SPECAC) y la suma del error de frecuencia en cada atributo en cada clúster (SEFAC), la suma de los dos valores forma la su total de error (STE). La siguiente prueba consiste en utilizar para la ejecución de los algoritmo la distancia euclidiana y la distancia haming, variando el valor de k. para ellos se va a tomar la SPECAC, la SEFA y la suma de los dos, estos valores se presentan en la tabla 87.

Tabla 87: Errores de los atributos particionales en Sisben21, variando el valor de k.

	Algoritmo	SPECAC	SEFAC	STE
K=2	**KMeans**	79650610852,104	14,148	79650610866,252
	K-Prototype	79650610852,104	14,148	79650610866,252
	K-Frequency	79650610852,104	14,148	79650610866,252
K=4	**KMeans**	562810645242,429	25,218	562810645267,646
	K-Prototype	562810645242,429	25,218	562810645267,646
	K-Frequency	562810645242,429	25,218	562810645267,646
K=5	**KMeans**	682245713958,703	32,086	682245713990,789
	K-Prototype	682245713958,703	32,086	682245713990,789
	K-Frequency	682245713958,703	32,086	682245713990,789
K=8	**KMeans**	681319410779,63	56,392	681319410836,022
	K-Prototype	681292260866,076	55,906	681292260921,982
	K-Frequency	681319410779,63	56,392	681319410836,022
K=16	**KMeans**	681041295214,596	94,635	681041295309,231
	K-Prototype	681176822122,285	114,666	681176822236,951
	K-Frequency	681209253824,966	112,674	681209253937,64

Analizando los resultados de los algoritmos de clustering particional se puede establecer que los atributos que más influyen en la formación de clústeres en datos con atributos mixtos, son los datos numéricos, debido a que muchas veces manejan escalas altas. También se puede establecer que al incrementar el valor

de k, los resultados de los algoritmos tienen mayor diferencia entre sí. También se puede establecer que no hay un algoritmo que sobresalga entre todos los demás, es decir no se encuentra un algoritmo que tenga un menor error en todos los valores de k.

8.4.2 Funcionalidad de los algoritmos de clustering particional con datos mixtos con k = 6.

El objetivo de la siguiente prueba es analizar los resultados de los algoritmos particionales. La prueba consiste dar como el valor de k el numero de niveles sisben que es 4 (k=4) en los algoritmos de tipo particional. Para ello se irán variando los tipos de distancia y se aplicara determinados filtros. Primero se ejecutan los algoritmos con los datos originales y se toma los representantes de estos clústeres (ver tabla 88) como parámetros de distancia, se utilizo la distancia euclidiana para los atributos numéricos y la distancia haming para los categóricos.

Tabla 88: Representantes de cada clúster formados con k=4 en el algoritmo kmeans utilizando la distancia eudidiana y haming.

C	z	v	p	b	ac	e	tv	tc	td	s	cn	tp	ec	sx	as	te	ne	a	ing	ed	ns	%
C0	1	2	4	1	1	2	3	4,17	2,87	4	4	4,14	2	1	2	0	2	1	1070486,99	45,06	3	0,49
C1	1	2	3	1	1	2	1	1,97	1,55	4	4	4,52	5	1	2	0	1	1	93413,56	37,54	1	18,53
C2	1	2	4	1	1	2	1	2,96	2,11	4	4	4,5	2	1	2	0	2	1	293863,4	40,72	2	7,12
C3	1	2	3	1	1	1	1	2,18	1,68	4	4	5,39	5	2	2	0	1	3	2205,53	21,62	1	73,86

Otra prueba consistió en utilizar la distancia mahalanobis para atributos numéricos, con la distancia haming para atributos categóricos. Los representantes de clústeres formados por los algoritmos se presentan en las tablas 89,90 y 91.

Tabla 89: Representantes de cada clúster formados con k=4 en el algoritmo kmeans utilizando la distancia mahalanobis y haming.

C	z	v	p	b	ac	e	tv	tc	td	s	cn	tp	ec	sx	as	te	ne	a	ing	ed	ns	%
C0	1	2	3	1	1	1	1	1,75	1,37	4	4	4,86	2	1	2	0	1	1	84901,9	39,97	1	21,01
C1	1	2	4	1	1	2	1	2,97	2,16	4	4	4,62	5	2	2	0	2	1	81855,27	30,45	2	29,02
C2	1	2	3	1	1	1	1	1,87	1,53	4	4	5,62	5	2	2	0	1	0	11628,67	22,52	1	28,94
C3	1	2	3	1	1	1	1	2,05	1,6	4	4	5,57	5	1	1	4	1	3	743,19	10,88	1	21,02

Tabla 90: Representantes de cada clúster formados con k=4 en el algoritmo kprototype utilizando la distancia mahalanobis.

C	z	v	p	b	ac	e	tv	tc	td	s	cn	tp	ec	sx	as	te	ne	a	ing	ed	ns	%
C0	1	2	3	1	1	1	1	1,6	1,19	4	4	5,31	5	2	2	0	1	1	30138,31	24,54	1	29,83
C1	1	2	3	1	1	1	3	3,23	2,55	4	4	6,81	5	2	2	0	1	1	41698,86	27,75	1	17,94
C2	1	2	4	1	1	2	1	3,15	2,24	4	4	4,85	5	2	2	0	1	1	74774,68	28,7	2	27,15
C3	1	1	3	1	2	2	1	1,16	1,08	4	4	4,13	5	2	2	0	1	1	33278,84	23,73	1	25,08

Tabla 91: Representantes de cada clúster formados con k=4 en el algoritmo kfrequency utilizando la distancia mahalanobis.

C	z	v	p	b	ac	e	tv	tc	td	s	cn	tp	ec	sx	as	te	ne	a	ing	ed	ns	%
C0	1	2	4	1	1	2	3	3,27	2,4	4	4	5,58	5	2	2	0	1	1	88578,64	34,12	2	31,36
C1	3	2	1	2	2	1	3	1,7	1,41	1	1	5,64	5	2	2	0	1	0	18960,57	27,25	1	15,9
C2	1	1	3	1	1	1	1	1,25	1,04	4	4	4,25	5	2	2	0	1	1	47135,17	28,10	1	29,78
C3	1	2	3	1	1	2	1	2,33	1,76	4	4	5,42	5	1	1	4	1	3	1280,78	11,50	1	22,97

Otra prueba consistió en utilizar la distancia mahalanobis para atributos numéricos y la proporción de coincidencias para atributos categóricos. Los representantes de clústeres formados por los algoritmos kmeans y kprototype se presentan en las tablas 92 y 93.

Tabla 92: Representantes de cada clúster formados con k=4 en el algoritmo kmeans utilizando la distancia mahalanobis y proporción de coincidencias.

C	z	v	p	b	ac	e	tv	tc	td	s	cn	tp	ec	sx	as	te	ne	a	ing	ed	ns	%
C0	1	1	3	1	1	1	1	1,2	0,99	4	4	3,36	5	2	2	0	1	1	40118,14	26,64	1	35,22
C1	1	2	4	1	1	2	3	4,33	3,33	4	4	6,41	5	2	2	0	1	1	75759,67	29,28	2	15,9
C2	1	2	3	1	1	2	1	2,65	2	4	4	5,58	5	2	2	0	1	1	48526	26,9	1	32,61
C3	1	2	3	1	1	1	1	1,41	0,99	4	4	6,99	5	1	2	0	1	3	19143,54	19,85	1	16,26

Tabla 93: Representantes de cada clúster formados con k=4 en el algoritmo kprototype utilizando la distancia mahalanobis y proporción de coincidencias.

C	z	v	p	b	ac	e	tv	tc	td	s	cn	tp	ec	sx	as	te	ne	a	ing	ed	ns	%
C0	1	2	4	1	1	2	3	4,33	3,34	4	4	6,37	5	2	2	0	1	1	75979,66	29,33	2	15,84
C1	1	2	4	1	1	2	1	2,76	1,99	4	4	4,3	5	2	2	0	1	1	60715,02	29,06	2	22,3
C2	1	2	3	1	1	1	3	2,43	1,99	4	4	8,46	5	2	2	0	1	0	22566,34	22,26	1	10,61
C3	1	1	3	1	1	1	1	1,26	0,99	4	4	4,47	5	2	2	0	1	1	33464,61	24,49	1	51,25

Otra prueba consistió en utilizar la distancia euclidiana para atributos numéricos y la distancia haming, para atributos categóricos pero aplicándolo a el conjunto de datos Sisben21 utilizando el filtro de normalizado. Los representantes de clústeres formados por los algoritmos kmeans, kprototype y kfrequency se presentan en las tablas 94, 95 y 96.

Tabla 94: Representantes de cada clúster formados con k=4 en el algoritmo kmeans utilizando la distancia euclidiana y haming en el conjunto de datos normalizado.

C	z	v	p	b	ac	e	tv	tc	td	s	cn	tp	ec	sx	as	te	ne	a	ing	ed	ns	%
C0	1	2	3	1	1	2	1	2,41	1,82	4	4	4,82	2	1	2	0	1	1	140962,3	39,25	1	26,6
C1	1	2	3	1	1	2	1	2,18	1,66	4	4	4,56	2	2	2	0	1	4	19183,96	42,1	1	21,08
C2	1	2	3	1	1	1	1	1,95	1,55	4	4	5,66	5	2	2	0	0	0	11635,07	13,92	1	28,28
C3	1	2	3	1	1	2	1	2,3	1,75	4	4	5,47	5	1	1	4	1	3	1167,42	11,6	1	24,04

Tabla 95: Representantes de cada clúster formados con k=4 en el algoritmo kprototype utilizando la distancia euclidiana y haming, en el conjunto de datos normalizado.

C	z	v	p	b	ac	e	tv	tc	td	s	cn	tp	ec	sx	as	te	ne	a	ing	ed	ns	%
C0	1	2	3	1	1	2	1	2,18	1,67	4	4	5,5	5	1	1	4	1	3	699,21	11,07	1	24,3
C1	1	2	3	1	1	2	1	1,58	1,31	4	4	4,79	5	2	2	0	1	1	38954,3	28,4	1	17,2
C2	1	2	3	1	1	1	1	1,86	1,5	4	4	5,48	5	2	2	0	1	1	36936,16	28,94	1	35,22
C3	1	2	4	1	1	2	1	3,2	2,28	4	4	4,58	5	2	2	0	1	1	108379,67	35,52	2	23,29

Tabla 96: Representantes de cada clúster formados con k=4 en el algoritmo kfrequency utilizando la distancia euclidiana, en el conjunto de datos normalizado.

C	z	v	p	b	ac	e	tv	tc	td	s	cn	tp	ec	sx	as	te	ne	a	ing	ed	ns	%
C0	1	2	3	1	1	2	1	2,32	1,76	4	4	5,42	5	1	1	4	1	3	1023,81	11,42	1	23,35
C1	1	2	4	1	1	2	1	2,98	2,15	4	4	5,48	5	2	2	0	1	1	85423,86	34,46	2	37,49
C2	3	2	1	2	2	1	3	1,72	1,43	1	1	5,75	5	2	2	0	1	0	18029,85	27,22	1	15,78
C3	1	1	3	1	2	1	1	1,17	1,07	4	4	3,98	5	2	2	0	1	1	42803,11	26,35	1	23,38

Otra prueba consistió en utilizar la distancia euclidiana para atributos numéricos y la proporción de coincidencias para atributos categóricos pero aplicándolo a el conjunto de datos normalizado. Los representantes de clústeres formados por los algoritmos kmeans y kprototype se presentan en las tablas 97 y 98

Tabla 97: Representantes de cada clúster formados con k=4 en el algoritmo kmeans utilizando la distancia euclidiana y la proporción de coincidencias en el conjunto de datos normalizado.

C	z	v	p	b	ac	e	tv	tc	td	s	cn	tp	ec	sx	as	te	ne	a	ing	ed	ns	%
C0	1	2	4	1	1	2	3	3,66	2,54	4	4	4,42	2	1	2	0	2	1	422486,66	43,4	2	4,57
C1	1	2	3	1	1	2	1	1,73	1,35	4	4	3,88	2	2	2	0	1	1	61256,69	44,13	1	32,1
C2	1	2	3	1	1	1	1	1,68	1,34	4	4	5,32	5	2	2	0	1	3	5171,36	11,79	1	45,94
C3	1	2	4	1	1	2	3	4,07	3,03	4	4	7,28	5	2	2	0	1	3	21720,33	25,76	1	17,39

Tabla 98: Representantes de cada clúster formados con k=4 en el algoritmo kprototype utilizando la distancia euclidiana y la proporción de coincidencias en el conjunto de datos normalizado.

C	z	v	p	b	ac	e	tv	tc	td	s	cn	tp	ec	sx	as	te	ne	a	ing	ed	ns	%
C0	1	2	3	1	1	1	1	2,29	1,84	4	4	9,25	5	2	2	0	1	0	14486,81	17,45	1	14,98
C1	1	2	3	1	1	1	1	1,6	1,27	4	4	4,36	5	2	2	0	1	3	8126,99	13,31	1	41,85
C2	1	2	4	1	1	2	3	4,49	3,29	4	4	5,72	5	2	2	0	1	1	88233,67	30,97	2	14,6
C3	1	2	3	1	1	2	1	1,88	1,41	4	4	3,89	2	2	2	0	1	1	93348,9	46,69	1	28,57

Otra prueba consistió aplicar el filtro R-frequency al conjunto de datos, este filtro convierte los atributos categóricos en numéricos, reemplazando el valor de la categoría por la frecuencia relativa que dicho valor tiene. Para esta prueba se utilizara solo el algoritmo kmeans utilizando la distancia eucliana, los representantes de los clústeres resultantes se presentan en la tabla 99.

Tabla 99: Representantes de cada clúster formados con k=4 en el algoritmo kmeans utilizando la distancia euclidiana en el conjunto de datos Sisben con filtro R-Frequency.

C	z	v	p	b	ac	e	tv	tc	td	s	cn	tp	ec	sx	as	te	ne	a	ing	ed	ns	%
C0	1	2	4	1	1	2	3	3,67	2,54	4	4	4,4	2	1	2	0	2	1	426797,93	43,51	2	4,47
C1	1	2	3	1	1	2	1	1,7	1,33	4	4	3,88	2	2	2	0	1	1	59292,59	43,11	1	33,8
C2	1	2	3	1	1	1	1	1,68	1,34	4	4	5,29	5	2	2	0	1	3	4108,65	11	1	43,21
C3	1	2	4	1	1	2	3	3,98	2,97	4	4	7,36	5	2	2	0	1	3	22884,39	25,76	1	18,52

Otra prueba consistió aplicar el filtro Convert al conjunto de datos, este filtro convierte los atributos categóricos en numéricos, reemplazando el valor de la categoría por un número entero. Para esta prueba se utilizara solo el algoritmo kmeans utilizando la distancia eucliana, los representantes de los clústeres resultantes se presentan en la tabla 100.

Tabla 100: Representantes de cada clúster formados con k=4 en el algoritmo kmeans utilizando la distancia euclidiana en el conjunto de datos Sisben con filtro Convert.

C	z	v	p	b	ac	e	tv	tc	td	s	cn	tp	ec	sx	as	te	ne	a	ing	ed	ns	%
C0	1	2	3	1	1	1	1	1,86	1,47	4	4	4,54	5	1	2	0	1	1	123189,61	38,39	1	23,1
C1	1	2	3	1	1	1	1	1,94	1,53	4	4	4,83	5	2	2	0	1	4	17413,46	39,83	1	25,34
C2	1	2	4	1	1	2	3	3,96	2,83	4	4	5,91	5	2	2	0	2	3	63416,41	23,38	2	18,7
C3	1	2	3	1	1	1	1	1,64	1,33	4	4	5,42	5	1	1	4	0	3	1213,95	8,25	1	32,87

Analizando todos los representantes obtenidos en las pruebas anteriores se puede establecer todos los clústeres formados son diferentes, que todos los algoritmos tienen en cuenta diferentes atributos, encontrando así diferentes tipos de similitud entre los elementos y creando clústeres con diferentes características. Entre todos los clústeres creados se puede destacar que en la mayoría de algoritmos se crea un cluster que tiene como resultado en el atributo nivel sisben el valor de categórico de 2 y el resto de los clústeres identificados se puede determinar que son 3 tipos de personas con diferentes características que tienen nivel sisben igual a 1. Entre las características que tienen los clústeres con nivel sisben igual a 2 se puede establecer que en los pisos, el material predominante es baldosa, vinilo, tableta o ladrillo y que tienen estrato 2. Otra característica se puede identificar es una alta relación en todos los clústeres formados entre el nivel de sisben el estrato.

8.5 Funcionalidad del algoritmo wayCluster.

La primera prueba consistió en aplicar el algoritmo wayCluster con un conjunto de datos numérico MUNDODES. Para el cual formo 53 clústeres, creando clúster con suma del error cuadrático igual a cero. La segunda prueba consistió en aplicar el algoritmo wayCluster con un conjunto de datos categórico ZooMamiferos, teniendo como resultado 13 clústeres, que tuvieron error de frecuencia igual a cero. Con el conjunto de datos sisben21 hubo problemas de espacio en memoria.

Analizando los resultados del algoritmo wayCluster, se deduce que la cantidad de clústeres formados es relativamente alta con respecto allá cantidad de registros a segmentar, pero se puede destacar que los clústeres formados tienen un alto grado de similitud.

9. CONCLUSIONES

Se presenta la primera versión de la herramienta "*RASEMUS: Una Herramienta para el Descubrimiento de Conocimiento en Bases de Datos con Técnicas de Clustering, Débilmente Acoplado con el SGBD PostgreSQL*".

La arquitectura de Rasemus es modular, compuesta por los módulos de interfaz grafica, kernel y utilidades, lo que permite la reutilización de sus componentes para incluirlos en otras herramientas de este tipo y facilita su mantenimiento.

Al estar esta herramienta desarrollada bajo un lenguaje multiplataforma, como lo es JAVA™, la convierte en una herramienta portable a cualquier sistema operativo.

Rasemus está elaborada bajo licencia GPL, aunque algunas API's utilitarias se encuentran bajo licenciamiento LGPL, como lo son iText y JFreeChart.

Las pruebas de funcionalidad realizadas con Rasemus confirman que esta herramienta soporta las etapas de minería de datos selección, preprocesamiento, clustering y visualización de resultados de una manera correcta y fiable.

La interfaz grafica de Rasemus es amigable, presentado un diseño atractivo y de fácil manejo, que permite exportar graficas y reportes de las características del conjunto de datos y de los resultados presentados por los algoritmos, en formatos pdf, rtf y html, que facilitan el análisis de informes.

En el kernel de Rasemus se encuentran implementados tres tipos de algoritmos de clustering como lo son: clustering particional, clustering jerárquico y clustering de densidad con diferentes tipos de medidas de distancia y similitud, con manejo de datos numéricos, categóricos y mixtos, que permiten al usuario escoger el algoritmo y la métrica que más se adecue a sus necesidades.

Se diseño e implemento el algoritmo kprototype propuesto por Ohn Mar San, Van-Nam Huynh y Yoshiteru Nakamori [20], el cual no ha sido implementado en ninguna herramienta analizada.

Se diseño e implemento dos nuevos algoritmos de clustering: el algoritmo wayCluster basado en la matriz de distancias, creando segmentos a partir de la unión de objetos con un alto grado de similitud, sin la necesidad de establecer el número de segmentos a formar y el algoritmo kfrequency basado en la utilización de la frecuencia relativa en los segmentos que se van formando como medida de similitud para atributos de tipo categórico, similar a él algoritmo KHistogram [20],

esto con el fin equiparar dicho valor, con los valores de las medidas de distancia de los atributos numéricos, haciendo funcional para los conjuntos de datos mixtos. Estos dos algoritmos son resultados importantes de esta investigación.

El análisis de las pruebas realizadas con los diferentes algoritmos de clustering, permite concluir que los algoritmos de tipo particional son eficientes con grandes cantidades de datos. Los algoritmos de tipo de datos jerárquico funcionan muy bien con repositorios de datos pequeños, que facilita la construcción de los dendogramas y el clustering de densidad sirve para crear segmentos cuando el usuario conoce el grado de similitud de los datos.

Basado en los mejores resultados obtenidos en las pruebas realizadas con conjuntos de datos con atributos de tipo numérico, es recomendable utilizar la distancia mahalanobis en los algoritmos de clustering.

El proyecto permitió afianzar los conocimientos adquiridos durante la carrera de Ingeniería de Sistemas e investigar y profundizar en una de las técnicas de descubrimiento de conocimiento muy importante como son las técnicas de clustering.

10. RECOMENDACIONES

Realizar otros tipos de pruebas para evaluar el rendimiento de Rasemus con respecto a otras herramientas que involucre la tarea de clustering, como el lenguaje R.

Hacer la implementación de kfrequency y wayCluster en el lenguaje R para evaluar su rendimiento y funcionalidad.

Utilizar Rasemus en proyectos de investigación que apliquen la tarea de minería de datos clustering, con conjuntos de datos reales, para soportar la toma de decisiones de una organización determinada.

Utilizar Rasemus en las electivas del área de Bases de Datos del programa de ingeniería de sistemas de la Universidad de Nariño, como herramienta didáctica y de apoyo.

11. REFERENCIAS BIBLIOGRÁFICAS

[1] Afify A., Pham D., Engineerring Aplications of Clustering Techniques. Manufacturing Engineering Centre, Cardiff University, Cardiff, UK.

[2] Agrawal R., Gehrke J., Gunopulos D. y Raghavan P., Automatic subspace clustering of high dimensional data for data mining applications. Pages 94–105, 1998.

[3] Agrawal R., Ghosh S., Imielinski T., Iyer B., Swami A., An interval Classifier for Database Mining Applications, VLDB Conference, Vancouver, Canada, 1992.

[4] Agrawal R., Srikant R., Fast Algorithms for Mining Association Rules, VLDB Conference, Santiago, Chile, 1994.

[5] Agrawal R., Srikant R., Mining Sequential Patterns. Proceedings of the 11th International Conference on Data Engineering, march, 1995.

[6] Agrawal R., Shim K., Developing Tightly-Coupled Data Mining Applications on a Relational and Database System, The Second International Conference on Knowledge Discovery and Data Mining, Portland, Oregon,1996

[7] Álvarez R., Estadística Multivariante y No Paramétrica con SPSS, Díaz Santos, 1994

[8] Anderberg M., Cluster Analysis For Applications,Acamenmic Press Inc. 1973

[9] Ankerst M., Breuning M., Kriegel H y J. Sander. Optics: Ordering points to identify clustering structure. In Proceedings of the ACM SIGMOD Conference, pages 49–60, Philadelphia, PA., 1999.

[10] Arias F., Maldonado H., FARC Terrorism in Colombia: A Clustering Analysis, Ministerio de Hacienda y Crédito Público, República de Colombia, enero 19, 2004.

[11] Babos A.,Iváncsy R., Legány C., Analysis and Extensions of Popular Clustering Algorithms. Department of Automatic and Applied Informatics and HAS-BUT Control Research Group Budapest University of Technology and Economics Goldmann Gy. Budapest, Hungary.

[12] Berry M., Linoff G., Data Mining Techniques for Marketing, Sales, and Customer Support, Wiley Computer Publishing, 1997.111

[13] Berry, M. y Linoff, G., Data Mining Techniques For Marketing, Sales, and Customer Relationship Management, 2da Edición, and Sons Inc, 2004.
[14] Bruera M., Segmentación: Un problema de Minería de datos y dos algoritmos,Data Management Division Software, Año 2, Número 5, 2001.

[15] Carmen M., Mora M., Browsing y Clustering: Dos Tecnicas en auge para la recuperación de información en línea. Documentación digital.Barcelona: Sección Científica de Ciencias de la Documentación del Departamento de Ciencias Politicas y Sociales de la Universidad de Pompeu Fabra, 2004.

[16] Cartagenova S., Detección automática de Reglas de Asociación, ITBA, 2005.

[17] Carpenter G. y Grossberg S., Art3: Hierarchical search using chemical transmiters in self-orginizing pattern recognition architectures. Neural Networks., 3:129–152, 1990.

[18] Chaudhuri S. Data Minino and Database Systems: Where is the Intersection?, Bulletin of the Technical Committe on Data Engineering, Vol 21 No. 1, Marzo, 1998.

[19] Chen M., Han J. y Yu P., Datamining: An Overview from Database Perspectiva, IEEE Transactions on KnowLed Data Engineering.1996

[20] Deng S., Dong B., He Z., Xu X., K-Histogram: An Efficient Clustering Algorithm for Categorical Dataset.

[21] Dzwinel W.,Yuen D., Boryczko K., Ben-Zion Y. , Yoshioka S., Ito T., Cluster Analysis, Data-Mining, Multi-dimensional Visualization of Earthquakes over Space, Time and Feature Space, Earth and Planetary Sci. Letters, , 2003

[22] Ester M., Krieguel H., Sander J y Xu X., A density-based algorithm for discovering clusters in large spatial databases, In KDD ’96: Conf. Knowledge Discovery Data Mining, pages 226–231, 1996.

[23] Fayyad U., Piatetsky-Shapiro G., Smyth P., From Data Mining to Knowledge Discovery: An Overview, in Advances in Knowlege Discovery and Data Mining, AAAI Pres/ The MIT Press, 1996.

[24] Fayyad U., Piatetsky-Shapiro G., Smyth P., The KDD Process for Extracting Useful Knowledge from Volumes of Data, Communications of the ACM, Vol. 39, No 11, November, 1996.

[25] Fayyad U., Piatetsky-Shapiro G., Smyth P., Knowledge Discovery and Data Mining: Towards a Unifying Framework, The Second International Conference on Knowledge Discovery and Data Mining, Portland, Oregon, 1996.

[26] Foss A., Lee C., Wang W., Zaïane O., On Data Clustering Analysis: Scalability, Constraints and Validation,

[27] G R. y Han J.. Very large data bases. In Proceedings of the 20th International Conference on Very Large Data Bases, pages 144–155, Berkeley, CA., 1994.

[28] González J., Morales E., Aprendizaje Computacional, Instituto Nacional de Astrofísica, Óptica y Electrónica

[29] Guha S., Rastogi R., y Shim K., Cure: an efficient clustering algorithm for large databases. In SIGMOD '98: Proceedings of the 1998 ACM SIGMOD international conference on Management of data, pages 73–84, New York, NY, USA, 1998. ACM Press.

[30] Guha S., Rastogi R y Shim K., ROCK: A robust clustering algorithm for categorical attributes. Information Systems, 25(5):345–366, 2000.

[31] Han J., Pei J., Minino Frequent Patterns without candidate generation, Proc. ACM SIGMOD, Dallas, TX,2000.

[32] Han J. y Kamber M.. Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers Inc, 2000.

[33] Hernández E. Algoritmo de clustering basado en entropía para descubrir grupos en atributos de tipo mixto, 2006

[34] Hernández J., Ramírez M. y Ferri C., Introducción a la Mineria de Datos, Prentice Hall, Madrid, 2004.

[35] Hertz J., Krogh A y Palmer R., Introduction to the theory of Neural computation. Addison-Wesley Longman Publishing Co, Inc., MA, Santa Fe Institute Studies in the Sciences of Complexity lecture notes, 1991.

[36] Hinneburg A. y Keim D., An efficient approach to clustering in large multimedia databases with noise. In Knowledge Discovery and Data Mining, pages 58–65, 1998.

[37] Huang Z. A fast clustering algorithm to cluster very large categorical data sets in data mining. In Research Issues on Data Mining and Knowledge Discovery, 1997.

[38] Huang Z. J., Clustering Categorical Data With K-Modes, Encyclopedia of Data Warehousing and Mining, Idea Group Inc (IGI), 2008

[39]Huang Z., Extensions to the k-means algorithm for clustering large data sets with categorical values. Data Mining and Knowledge Discovery, 1998.

[40] Huynh V., Nakamori Y., San O., An Alternative Extension of the K-Means Algorithm For Clustering Categorical Data, Int. J. Appl. Math. Comput. Sci., 2004, Vol. 14, No. 2, 241–247

[41] Imielnski T., Mannila, H., A Database Perspectiva on Knowledge Discovery, Comunications of the ACM, Vol 39, No.11, November, 1996

[42] Inza I., Larrañaga P., Moujahid A., Clustering, Departamento de Ciencias de la Computación e Inteligencia Artificial, Universdad del País Vasco-Euskal Herriko Unibertsitatea.

[43] Jain A. K., Murty M.N., y Flynn P. J.. Data clustering: a review. ACM Computing Surveys, 31(3):264–323, 1999.

[44] JAIN, A. and DUBES, R. 1988. Algorithms for Clustering Data. Prentice-Hall, Englewood. Cliffs, NJ

[45] Karypis G., Han y Kumar V., NEWS. Chameleon: Hierarchical clustering using dynamic modeling. Computer, 32(8):68–75, 1999.

[46] Kauffman L. y Rousseuw P., Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley and Sons, New York, NY, 1990.

[47] Kohonen T., Self-orgniztion and assosiative Memory. Springer information sciencesseries, Springer-Verlag, NY, third edition, 1989.

[48] Lee G., Lee W., Li M., Sivasubramaniam A., PENS: An Algorithm for Density–Based Clustering in Peer-to-Peer Systems.

[49] Mercer D., College L., Clustering Large Datasets, 2003

[50] Milenova B. y Campos M., Clustering large databases with numeric and nominal values using orthogonal projections. In Proc. of the ACM SIGMOD Conf. Management of Data, 2002.
[51] Nueller C., Data Clustering, 2005
[52] Navas M., Un modelo de clustering temporal, 2004

[53] Nong Ye., The Handbook Of Data Mining, Lawrence Erlbaum Associates, Inc. 2003.

[54] Parr O., Data Mining Cookbook Modeling Data for Marketing, Risk, and Customer Relationship Management, John Wiley & Sons, Inc. 2001.

[55] Pascual D., Pla F., Sánchez S., Algoritmos de Agrupamiento.

[56] Pawlak Z., Rough sets: Theoretical aspects of reasoning about data, Kluwer Academic Publishers, 1991.

[57] Peña D., Analisis de Datos Multivariantes, McGrawHill, España 2002

[58] Pervesi I., Aplicación de Minería de Datos Para la Exploración y Detección de Patrones Delictivos en Argentina. Instituto Tecnológico de Buenos Aires. 2007.

[59] Piatetsky-Shapiro G., Brachman R., Khabaza T., An Overview of Issues in Developing Industrial Data Mining and Knowledge Discovery Applications, 1996

[60] Rodriguez D., Analisis de Datos de Expresión Genetica Mediante Técnicas de Bioclustering.2006.

[61] Sheikholeslami G., Chatterjee S. y Zhang A., WaveCluster: A multi-resolution clustering approach for very large spatial databases. In Proc. 24th Int. Conf. Very Large Data Bases, VLDB, pages 428–439, 24–27 1998.

[62] Srikant R., Agrawal R., Mining Quantitative Association Rules in Large Relational Tables, ACM SIGMOD, Montreal, Canada, 1996.

[63] Shan N., Ziarko W., Hamilton H., Cercone N., Discovering Classification Knowledge in Databases using Rough Sets, The Second International Conference on Knowledge Discovery and Data Mining, Portland, Oregon, 1996.

[64] Selim S., Ismail M., K-Means-type algorithms: A generalized convergence theorem and characterization of local optimality. IEEE Trans. Pattern Anal. Mach. Intell., Vol. PAMI-6, No. 1, pp. 81–87. 1984.

[65] Timáran, R., Arquitecturas de Integración del Proceso de Descubrimiento de Conocimiento con Sistemas de Gestión de bases de satos: un estado del Arte, en

revista Ingeniria y Competitividad, Univeridad del Valle, Volumen 3, No. 2, Cali, 2001.
[66] Wang W., Yang J y Muntz R., Sting: a statistical information grid approach to spatial data mining. In Proceedings of the 23rd Conference on VLDB, pages 186–195, Athens, Greece, 1997.

[67] Zadeh L. Fuzzy sets. Inf. Control, 8:338–353, 1965.

[68] Zhang T., Ramakrishnan R y Livny Mi., Birch: an efficient data clustering method for very large databases. In SIGMOD '96: Proceedings of the 1996 ACM SIGMOD international conference on Management of data, pages 103–114, New York, NY, USA, 1996. ACM Press.

[69] Zhou W., A Review and Implementation of Some Approaches to Metrics Clustering. 2004

[70] http://www.alphaminer.org/

[71] http://www.biomedcentral.com/1471-2105/8/S7/S17/

[72] http://databionic-esom.sourceforge.net/

[73] http://www.dbminer.com

[74] http://www.definicionabc.com/geografia/distancia.php

[75] http://datamining.itsc.uah.edu/adam/

[76] http://clusty.com/

[77] http://glaros.dtc.umn.edu/gkhome/views/cluto

[78] http://www.gnu.org/software/pspp/

[79] http://www.gsi.dit.upm.es/~anto/tesis/html/stateart.html

[80] http://www.gisdevelopment.net/technology/ip/techip009.htm

[81] http://www.google.com/support/news/bin/answer.py?hl=en&answer=40235

[82]http://www-01.ibm.com/software/data/iminer/

[83] http://jexcelapi.sourceforge.net/

[84]http://infobiochip.isciii.es/Textos/Metodologia/bioinfo%20asociada/metodologia/analisis%20de%20datos.htm
[85] http://www.itl.nist.gov/div897/sqg/dads/HTML/manhattanDistance.html

[86] http://kdd.ics.uci.edu/

[87] http://www.keel.es/

[88] http://www.knime.org/

[89] http://lejarza.ecoapl.uv.es/multivari/cluster/CLUSTER.htm

[90] http://www.lowagie.com/iText/

[91] Orange http://magix.fri.uni-lj.si/orange/

[92] http://msdn.microsoft.com/es-es/library/ms175609(SQL.90).aspx

[93] http://www.netbeans.org/kb/60/java/quickstart-gui-legend.html

[94] http://www.oracle.com/technology/products/bi/odm/index.html

[95] http://www.lirmm.fr/~caraux/PermutMatrix/

[96] http://rapid-i.com/content/view/113/136/lang,en/

[97] http://www.rational.com.ar/brochures/db2im.pdf

[98] http://www.spss.com/

[99] http://tariykdd.berlios.de/

[100] http://www.cs.waikato.ac.nz/ml/weka/

[101] http://es.wikipedia.org/wiki/Distancia

[102] http://es.wikipedia.org/wiki/Distancia_euclidiana

[103] http://es.wikipedia.org/wiki/Distancia_de_Mahalanobis

[104] http://es.wikipedia.org/wiki/Distancia_Hamming

[105] http://es.wikipedia.org/wiki/IText.

[106] http://archive.ics.uci.edu/ml/

ANEXOS

Anexo 1. CONJUNTO DE DATOS UDENAR

Discretización conjunto de datos UDENAR.

Tabla 1. Discretización atributo Edad

EDAD	***DISCRETIZACIÓN***	**CANTIDAD**
Menores e iguales a 18	A	827
Mayores de 18 y menores que 22	B	3634
Mayores e iguales que 22 y Menores de 26	C	4856
Mayores e iguales que 26	D	11012

Tabla 2. Discretización atributo Edad_ingreso

EDAD DE INGRESO	**DISCRETIZACIÓN**	**CANTIDAD**
Menores e iguales a 18	A	9054
Mayores de 18 y menores que 22	B	7370
Mayores e iguales que 22 y Menores de 26	C	2594
Mayores e iguales que 26	D	1311

Tabla 3. Discretización atributo Fecha ingreso

FECHA DE INGRESO	**DISCRETIZACIÓN**	**CANTIDAD**
Antes de 1990	A	1022
Después o igual a 1990 a Menores de 1995	B	4852
Después o igual a 1995 a Menores de 2000	C	5978
Después o igual al 2000 y Menores de 2003	D	5046
Mayores o iguales de 2003	E	7621

Tabla 4. Discretización atributo Ingresos

INGRESOS	DISCRETIZACIÓN	CANTIDAD
Menores q 30000000	A	6101
Mayores = 3 y meno 6	B	6350
Mayor de =6 y menor q 9	C	3325
Mayor =9 y menores q 12	D	1828
Mayores = 12	E	2725

Tabla 5. Discretización atributo Valor Matricula

VALOR DE MATRICULA	DISCRETIZACIÓN	CANTIDAD
Menores q 50000	A	3529
Mayor = 50000 y menor 100000	B	5348
Mayor = 100000 y menor 200000	C	6762
Mayor = 200000 y menor q 500000	D	3636
Mayores q = 500000	E	1054

Tabla 6. Discretización atributo Naturaleza

NATURALEZA	*DISCRETIZACIÓN*	*CANTIDAD*
OFICIAL	OFICIAL	13571
NO OFICIAL	PRIVADO	6757

Tabla 7. Discretización atributo Ocupacion Madre

OCUPACION MADRE	*DISCRETIZACIÓN*	*CANTIDAD*
AMA DE CASA	1	9049
PROFESIONAL	2	572
EMPLEADA	3	313
INDEPENDIENTE	4	200
OTRO	5	10194

Tabla 8. Discretización atributo Ponderado

PONDERADO	DISCRETIZACIÓN	CANTIDAD
Menores a 30	A	43
Mayores e iguales que 30 y menores que 50	B	2689
Mayores e iguales que 50 y menores que 70	C	16601
Mayores e iguales que 70 y menores e iguales que 100	D	1623

Tabla 9. Discretización atributo Estado Civil

ESTADO CIVIL	DISCRETIZACIÓN	CANTIDAD
Soltero	0	20651
Casado	1	5
Separado	2	251
Divorciado	3	24
Unión Libre	4	25

Tabla 10. Discretización atributo Tipo Residencia

TIPO_RESIDENCIA	DISCRETIZACIÓN	CANTIDAD
No propia	0	5181
Propia	1	14995
Propia pagando cuotas	2	308
Arrendando o Anticresis	3	472

Tabla 11. Discretización atributo Semestre

SEMESTE	*DISCRETIZACIÓN*	*CANTIDAD*
Semestre 1 hasta semestre 4	1	8859
Semestre 5 hasta semestre 8	2	3539
Semestre 9 hasta semestre 10	3	1390
Semestre (11–15) (egresados) – 20 (Demorados en egresar))	4	1383
Semestre 110 (graduados)	5	5157

Tabla 12. Discretización atributo Facultad

FACULTAD	DISCRETIZACIÓN	CANTIDAD
ARTES	1	2097
CIENCIAS AGRICOLAS	2	1532
DERECHO	3	1208
CIENCIAS ECONOMICAS Y ADMINISTRATIVAS	4	2424
INGENIERIA	5	2549
CIENCIAS PECUARIAS	6	1715
CIENCIAS NATURALES Y MATEMATICAS	7	3701
CIENCIAS HUMANAS	8	4009
EDUCACION	18	793
INGENIERIA AGROINDUSTRIAL	22	537
CIENCIAS DE LA SALUD	32	390

Tabla 13. Discretización atributo Clase_al

CLASE_AL	DISCRETIZACIÓN	CANTIDAD
Normal	1	18641
Reingreso	2	846
Retirado	3	1469

Tabla 14. Discretización atributo Clase_rend

CLASE_REND	DISCRETIZACIÓN	CANTIDAD
No ha perdido materias	1	7852
Ha perdido 1 materia	2	3339
Ha perdido 2 materia	3	2576
Ha perdido 3 materia	4	2047
Ha perdido mas de 3 materias	5	5142

Tabla 15. Discretización atributo Clase_promedio

CLASE_PROMEDIO	DISCRETIZACIÓN	CANTIDAD
Menor a 2	A	2391
Mayor o igual a 2 hasta 3	B	2934

CLASE_PROMEDIO	DISCRETIZACIÓN	CANTIDAD
Mayor o igual a 3 hasta 3.5	C	5166
Mayor o igual a 3,5 hasta 4,0	D	6850
Mayor o igual a 4.0 hasta 5.0	E	3615

Tabla 16. Discretizacion atributo Zona_nac

ZONA_NAC	DISCRETIZACIÓN	CANTIDAD
Norte del departamento de Nariño	NORTEN	66
Sur del departamento de Nariño	SURN	16931
Oriente del departamento de Nariño	ORIENTEN	1512
Occidente del departamento de Nariño	OCCIDENTENN	595
Norte de Colombia	NORTE	62
Sur de Colombia	SUR	410
Oriente de Colombia	ORIENTE	23
Occidente de Colombia	OCCIDENTE	729

Anexo 2. CONJUNTO DE DATOS SISBEN

Atributo	Valores	Descripción
zona	1. Cabecera 2. Centro poblado 3. Rural disperso	Que tipo de zona de asentamiento es.
vivienda	1. Cuartos(s) en casa o apartamento 2. Casa o apartamento 3. Otro tipo de unidad de vivivenda	Tipo de unidad de vivienda
piso	1.Tierra o arena 2 .Madera burda, tabla o tablón 3. Cemento o gravilla 4. Baldosa, vinilo, tableta o ladrillo 5. Alfombra	Material predominante en pisos
basura	1.Si 2.No	Servicio de recolección de basura.
acueducto	1.Si 2.No	Servicio de acueducto.
estrato	0,1,2,3,4	Estrato del recibo de luz
teneviv	1. Arriendo o subarriendo 2. Propia pagando 3. Propia pagada 4. Otra condicion	La vivienda en la que viven es
tcuartos	numerico	Cantidad de cuartos de la casa
tdormir	numérico	Cantidad de cuartos para dormir
sanitar	0. No tiene 1. Letrina, bajamar 2. Inodoro sin conexión alcantarrillado ni a pozo séptico 3. Inodoro con conexión a poso séptico 4.Inodoro con conexión a alcantarillado	Servicio de sanitario que usan
cocinan	0. No cocinan 1. Leña, carbón de leña, desechos 2. Carbón mineral 3. Kerosene, petróleo, gasolina, cocinol 4. Gas en cilindro o pipeta 5. Gas con conexión por tubería 6. Electricidad	Combustible con el que cocinan
tpersonas	numérico	Número total de personas que conforman la familia
estcivil	1. Union libre 2. Casado	Estado civil

	3. Viudo 4. Separado o divorciado 5. Soltero	
sexo	1. Hombre 2. Mujer	Sexo o genero
asiste	1. Si 2. No	Asiste a un centro educativo
tipoesta	0. Ninguno 1. Centros de atención u hogares ICBF 2. Guarderia, salacuna, preescolar, jardín infantil oficial 3. Guarderia, salacuna, preescolar, jardín infantil no oficial 4. Escuela, colegio, técnico universitario o universidad oficial 5. Escuela, colegio, técnico universitario o universidad no oficial 6. Sena 7. Secundaria técnica oficial 8. Secundaria técnica no oficial	Tipo de establecimiento educativo
nivel	0. Ninguno 1. Primaria 2. Secuandria 3. Técnica o tecnológica 4. Universidad 5. Postgrado	Nivel educativo
activi	0. Sin actividad 1. Trabajando 2. Buscando trabajo 3. Estudiando 4. Oficios del hogar 5. Rentista 6. Jubilado, pensiando 7. Inválido	Actividad en el último mes
ingresos	numérico	Ingresos mensuales
edad	numérico	edad
NIVSIS	1, 2, 3, 4	Nivel sisben asignado

Anexo 3. MANUAL DE USUARIO

RASEMUS V 1.0

Rasemus es una herramienta débilmente acoplada con el sistema gestor de base de datos PostgreSQL con Licencia Pública General de GNU (GPL) orientada principalmente a proteger la libre distribución, modificación y uso. Desarrollada por Ricardo Narváez Terán integrante del grupo de investigación GRIAS (Grupo de Investigación Aplicado a Sistemas), Dirigido por el Dr. Ricardo Timarán Pereira Ph.D, del programa de ingeniería de sistemas de la Universidad de Nariño de la ciudad de San Juan de Pasto (Nariño - Colombia).

La palabra Rasemus es una palabra del latín que significa racimo o grupo.

Rasemus es un software de apoyo a la toma de decisiones, mediante la ejecución de algoritmos de clustering.

Rasemus le permite al usuario escoger diferentes fuentes de datos como lo son archivos con formato .ARFF, formato .XLS, archivos planos y el sistema gestor de base de datos PostgreSQL. Permite utilizar filtros para la limpieza, selección y transformación del conjunto de datos. Rasemus le da la posibilidad de utilizar 3 técnicas de clustering que son: la técnica de clustering particional, con los algoritmos kmeans, kprototype y kfrequency, la técnica de clustering jerárquico con tres tipos de enlace: enlace mínimo (Single-link), enlace promedio (Averange-link) y enlace Maximo (Complete-link) y la técnica de clustering basada en densidad con el algoritmo DBScan con sus variaciones barrido directo () y barrido conectado. La presentación de los resultados se puede ver informes textuales que resumen todo el comportamiento de los algoritmos permitiendo su exportación a formatos .PDF, .HTML y .RTF, también utilizando graficas de dispersión para la visualización de los segmentos formados y la visualización del dendograma resultado de los algoritmos jerárquicos y la posibilidad de exportar las estas imágenes.

Su interfaz grafica es amigable, diseñada con la técnica Drag & Drop para la realización del flujo de trabajo, permitiendo a los usuarios construir el proceso de selección de las operaciones necesarias de forma ágil. Además su manejo de colores hace que Rasemus tenga una presentación moderna de sus iconos, acorde a la tendencia actual.

REQUERIMIENTOS

Lo único que se necesita para el correcto funcionamiento de la herramienta en cualquier sistema operativo, es tener instalado la maquina virtual de Java 1.6 o superior.

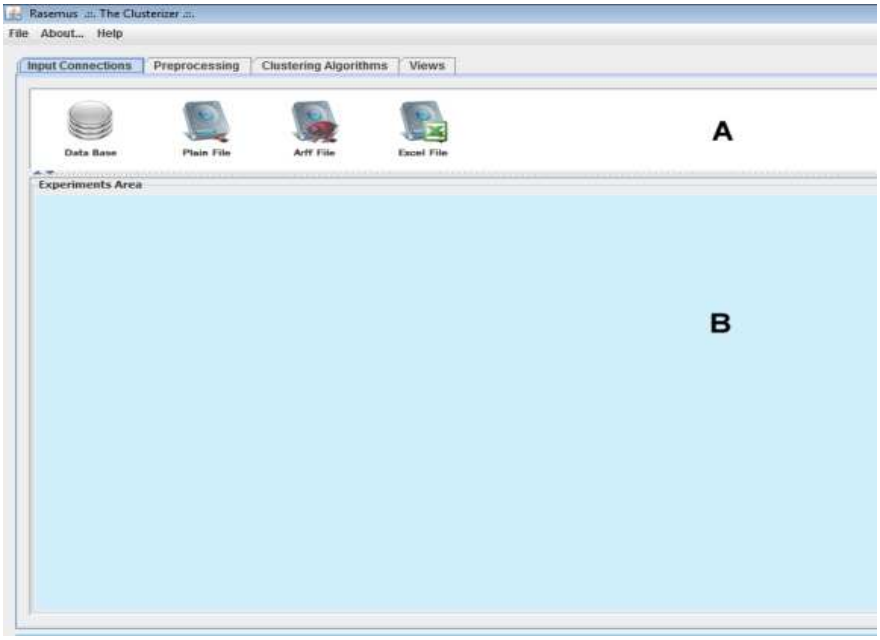
Nota: En las versiones anteriores de la maquina virtual pueden perderse algunos elementos.

1. EMPEZAR A UTILIZAR RASEMUS

Rasemus es desarrollo bajo el leguaje de programación Java, lo que le permite ser multiplataforma, para ello Rasemus se presenta en un archivo .jar. para no perder la opción de ser una herramienta multiplataforma. Se toma el archivo .jar y se da doble click.

La ventana principal está compuesta por dos sectores. En la figura 1 se muestra la ventana principal en donde se identifican dos zonas. La zona A es la parte en donde se presentan las tareas que se necesitan para realizar las tareas de descubrimiento de conocimiento, representadas en iconos, divididos en pestañas según su especialidad. En la parte B, está el área de trabajo, en donde se colocan los iconos que el usuario elija trabajar.

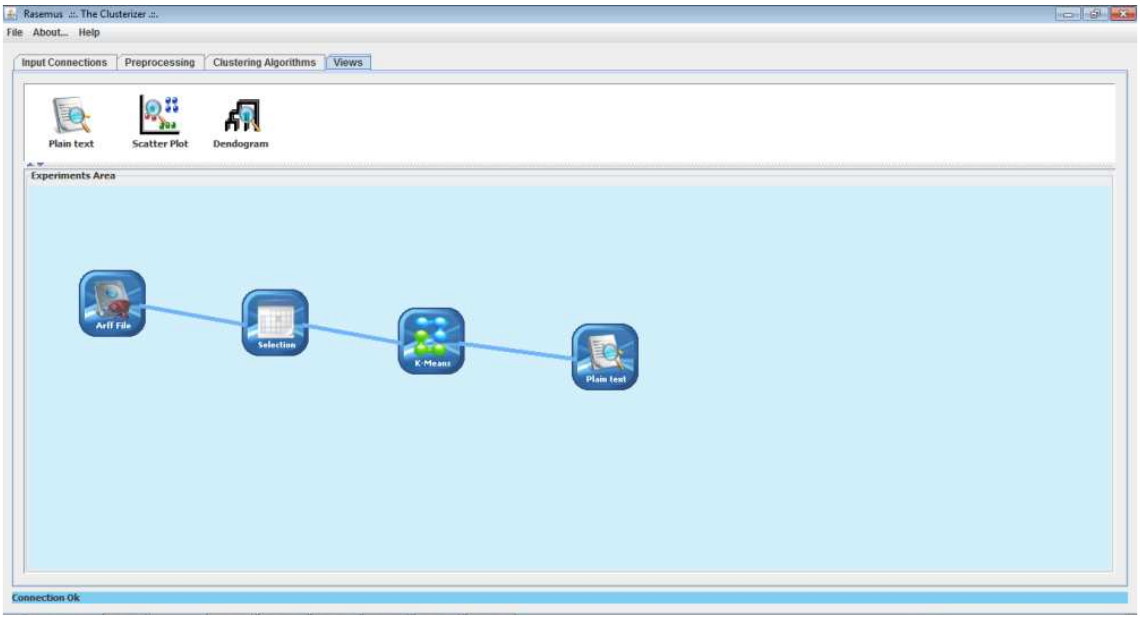
Figura 1: Ventana principal



Para la colocación de los iconos de las tareas en el área de trabajo se debe dar click sostenido sobre el icono y arrástralo hasta el lugar del área de trabajo donde desee colocar el icono. Para unir los iconos y crear flujos de trabajos, los iconos están equiparados con dos áreas de conexión, una en la parte derecha, y otra en la parte izquierda, que funcionan de la siguiente manera: primero se da click sostenido en el área del icono que se decida como fuente, sin soltar el botón derecho del mouse, se arrastra el puntero hasta el área de conexión del icono destino y se suelta el botón y se establece la conexión.

Figura 2: Forma de Conexión

Figura 3: Manejo de la operaciones.



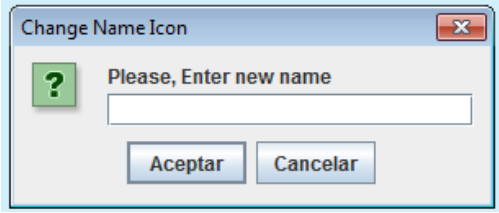
Los iconos ubicados en el área de trabajo son configurados, colocando el puntero del mouse sobre el icono y dando click derecho. Ahí se despliega un menú en donde se presentan las diferentes opciones de configuración dependiendo del tipo de icono seleccionado (ver figura 4). Los iconos tienen como opciones básicas:

Delete icon
Permite borrar el icono

Figura 4: Icono de Rasemus.

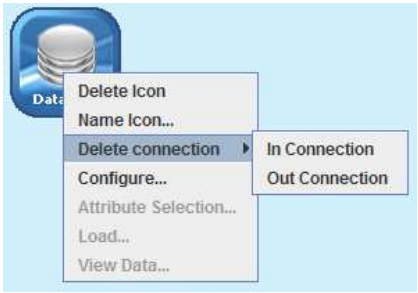
Name Icon
Permite cambiar el nombre del icono, desplegando una ventana emergente (ver figura 5)

Figura 5: Opción Name Icon.



Name Icon
Permite borrar las conexiones del icono, esta opción despliega un submenú en donde se puede establecer qué tipo de conexiones quiere borrar, las entrantes con *In Connections* o las salientes con *Out Connections.* Ver figura 6.

Figura 6: Opción Delete connections.



1.1 Input Connections o Selección Fuente de Datos:

En la pestaña Input Connections, se presentan los diferentes iconos que permiten la selección de fuentes de datos, entre ellas está el icono Data Base, que permite la conexión con el sistema gestor de bases de datos PostgreSQL, el icono Plain File, que permite la conexión con tipos de datos con formato plano, también se encuentra el icono Arff File que permite la conexión con archivos de formato .Arff y por ultimo esta el icono Excel File, que permite la conexión con archivos con formato .Xls. ver figura 7.

Figura 7: Iconos de la opción Input Connections.

1.1.1 Icono Data Base

Permite la conexión con el sistema gestor de bases de datos PostgreSQL, con las opciones que se presenta en la figura 8.

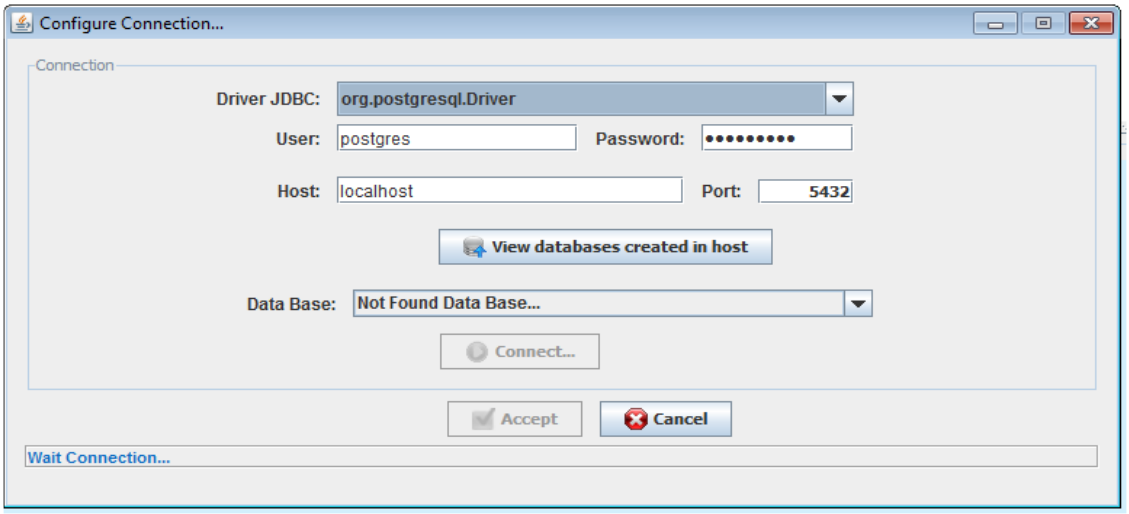
Figura 8: Opciones Icono Data Base.

Configure

La opción configure despliega una ventana que solicita los parámetros básicos para la establecer la conexión a una base de datos (ver figura 9), como el usuario, el password, el servidor y el puerto. Ya configurados estos elementos se presiona

el botón *View Found created in host,* que lista las bases de datos creadas en el servidor, permitiéndole al usuario escoger la base de datos que desea trabajar. Por último se presiona el botón *Connect* y si los datos son correctos, en la barra de estado que está en la parte inferior presenta un mensaje OK; si los datos suministrados no son correctos se presenta el mensaje de error correspondiente. Si la conexión es correcta, para finalizar se presiona el botón *Accept*. Se habilita la opción configure.

Figura 9: Ventana de configuración de conexión a base de datos.



***Attribute Selection*:**

La opción *Attribute selection* permite seleccionar los atributos de las tablas que se encuentran en a base de datos, las tablas de la base de datos se presentan en un menú desplegable ubicado en la parte superior de la ventana, el usuario elije la tabla a trabajar y se adiciona al área de trabajo. la ventana se encuentra separada en tres partes (ver figura 10):

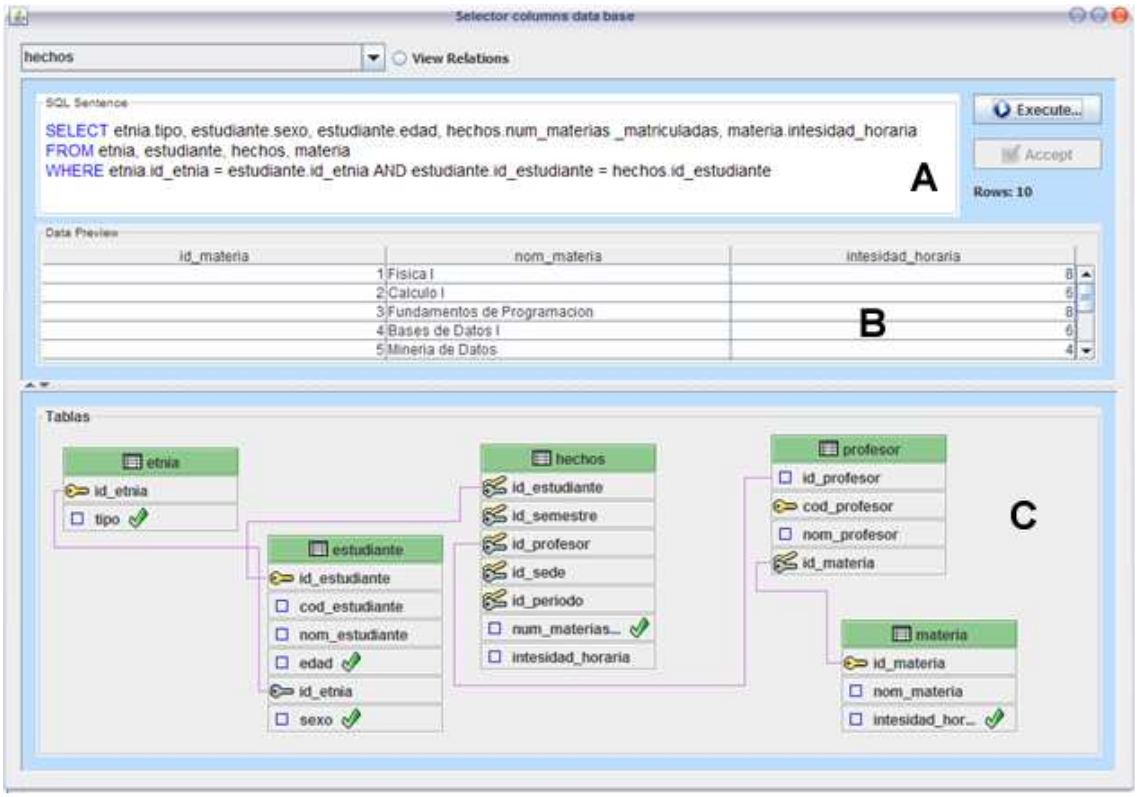
A) área de visualización del sql, en donde se va mostrando la construcción de la sentencia sql, con los diferentes atributo seleccionados. Aquí se encuentran los botones *Execute* que permite ejecutar la sentencia sql formado, y hace visualizar lo resultados en la parte B.

B) en esta zona se presenta el resultado de la sentencia sql formada.

C) En esta zona se encuentran las tablas seleccionadas por el usuario. Presenta las tablas como objetos independientes dando la posibilidad que con click sostenido sobre su cabecera pueda ubicarla en cualquier lugar de su área. Para selección el atributo que se desea trabajar se debe dar click sobre la parte izquierda del atributo. En el área de trabajo también observan las llaves primarias y foráneas de las tablas, permitiendo también ver una línea de relación entre las referencias de las tablas que facilitan la labor de la selección

de atributos.

Figura 10: Ventana de selección de atributos.



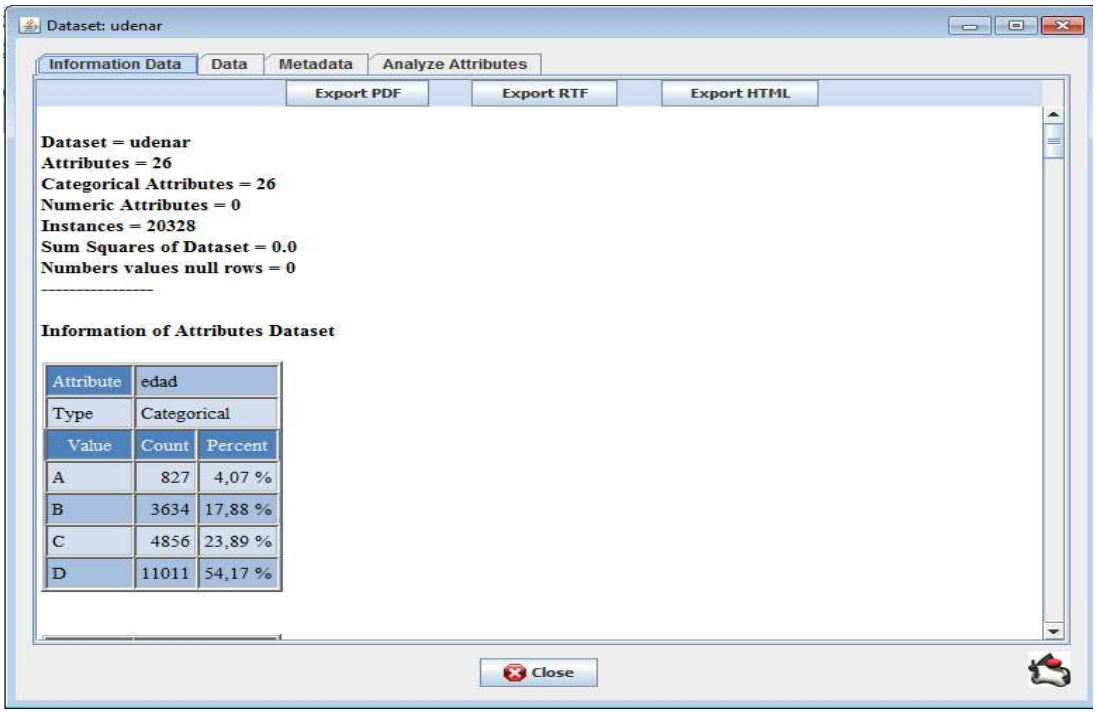
Load

Permite cargar los datos de los atributos seleccionados a memoria. Sin esta opción no se puede conectar el icono con ningún otro icono.

View Data

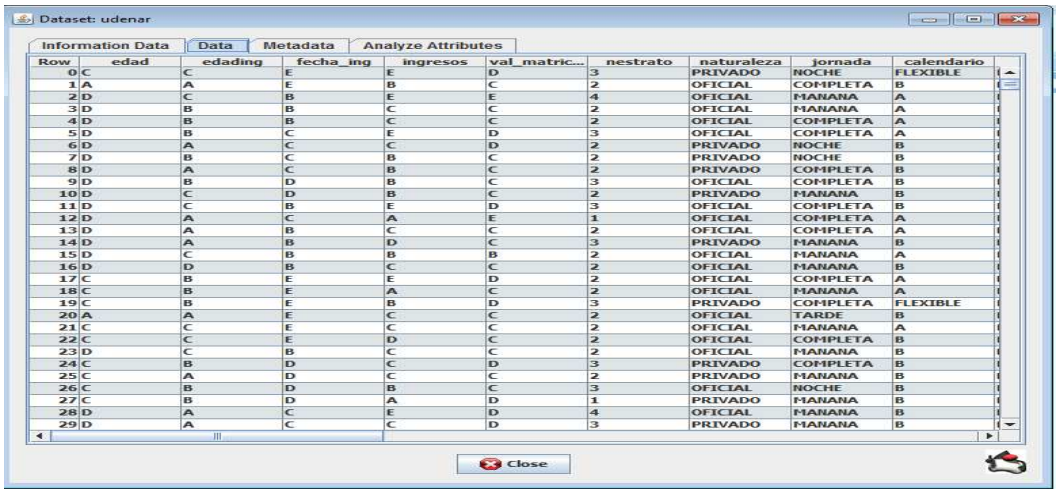
Permite mirar información acerca del conjunto de datos. En la primera pestaña *Information Data* se presenta las características del conjunto de datos y sus atributos, esta información se puede exportar a archivos con formato .PDf, .RTF y .HTML.(ver figura 11)

Figura 11: Pestaña *Information Data* opción View Data.



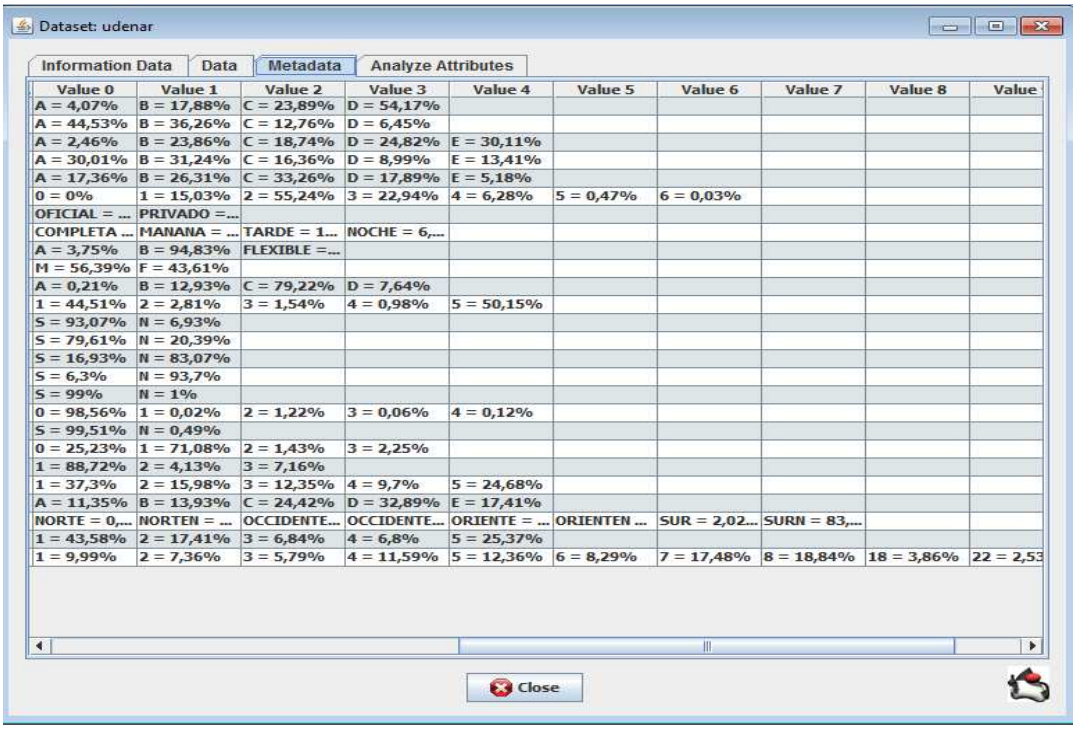
La segunda pestaña denominada *Data* presenta los registros del conjunto de datos. Aquí se puede ordenar el por atributo los datos dando click sobre el nombre de la columna. Ver figura 12.

Figura 12: Pestaña *Data* opción View Data.



La tercera pestaña denominada *Metadata* presenta las características de los atributos, para el caso de los atributos presenta el valor mínimo, la media, el valor máximo y mucho mas, para los atributos de tipo categórico se presenta sus valores con sus respectivas frecuencias absolutas y porcentaje en el conjunto de datos. Esta información le permite comprender mejor sus datos de sus datos, ver figura 13.

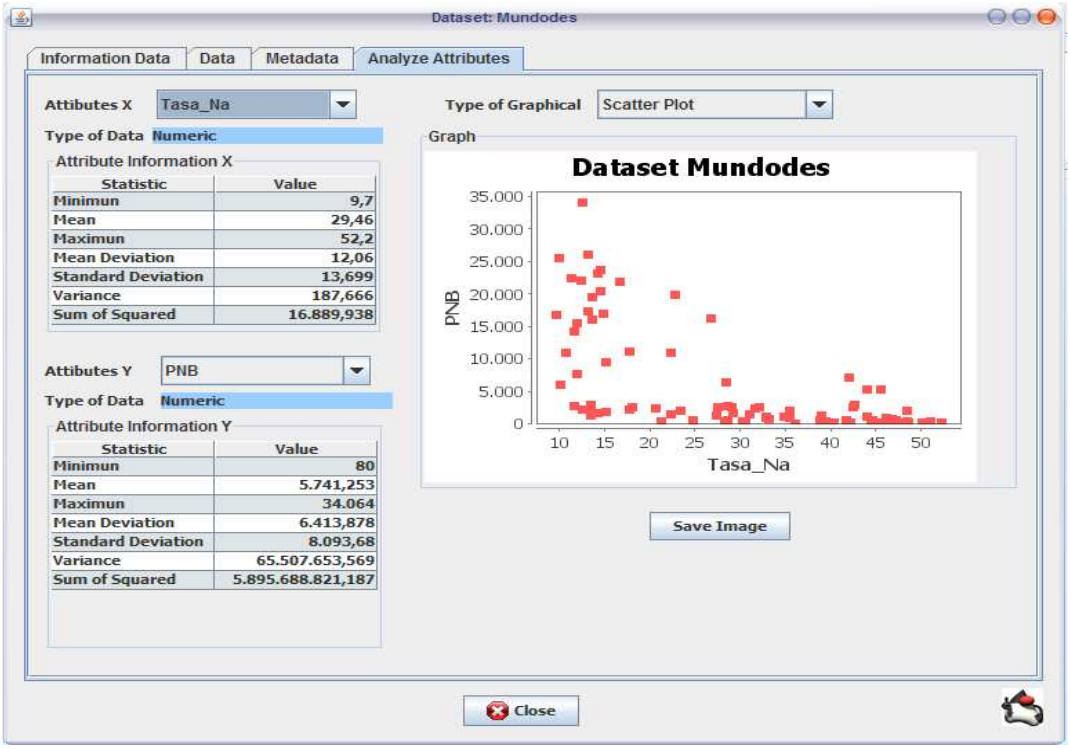
Figura 13: pestaña *Metadata* opción View Data.



Value 0	Value 1	Value 2	Value 3	Value 4	Value 5	Value 6	Value 7	Value 8	Value
A = 4,07%	B = 17,88%	C = 23,89%	D = 54,17%						
A = 44,53%	B = 36,26%	C = 12,76%	D = 6,45%						
A = 2,46%	B = 23,86%	C = 18,74%	D = 24,82%	E = 30,11%					
A = 30,01%	B = 31,24%	C = 16,36%	D = 8,99%	E = 13,41%					
A = 17,36%	B = 26,31%	C = 33,26%	D = 17,89%	E = 5,18%					
0 = 0%	1 = 15,03%	2 = 55,24%	3 = 22,94%	4 = 6,28%	5 = 0,47%	6 = 0,03%			
OFICIAL = ...	PRIVADO =...								
COMPLETA ...	MANANA = ...	TARDE = 1...	NOCHE = 6,...						
A = 3,75%	B = 94,83%	FLEXIBLE =...							
M = 56,39%	F = 43,61%								
A = 0,21%	B = 12,93%	C = 79,22%	D = 7,64%						
1 = 44,51%	2 = 2,81%	3 = 1,54%	4 = 0,98%	5 = 50,15%					
S = 93,07%	N = 6,93%								
S = 79,61%	N = 20,39%								
S = 16,93%	N = 83,07%								
S = 6,3%	N = 93,7%								
S = 99%	N = 1%								
0 = 98,56%	1 = 0,02%	2 = 1,22%	3 = 0,06%	4 = 0,12%					
S = 99,51%	N = 0,49%								
0 = 25,23%	1 = 71,08%	2 = 1,43%	3 = 2,25%						
1 = 88,72%	2 = 4,13%	3 = 7,16%							
1 = 37,3%	2 = 15,98%	3 = 12,35%	4 = 9,7%	5 = 24,68%					
A = 11,35%	B = 13,93%	C = 24,42%	D = 32,89%	E = 17,41%					
NORTE = 0,...	NORTEN = ...	OCCIDENTE...	OCCIDENTE...	ORIENTE = ...	ORIENTEN ...	SUR = 2,02...	SURN = 83,...		
1 = 43,58%	2 = 17,41%	3 = 6,84%	4 = 6,8%	5 = 25,37%					
1 = 9,99%	2 = 7,36%	3 = 5,79%	4 = 11,59%	5 = 12,36%	6 = 8,29%	7 = 17,48%	8 = 18,84%	18 = 3,86%	22 = 2,53

La cuarta pestaña denominada *Analyze attributes* presenta las características de los atributos. Dando la opción de crear graficas de diferente tipo como diagramas de dispersión, si es atributo numérico o diagramas de pastel si son categóricos y demás graficas. Estas graficas pueden ser exportadas como JPG, presionando el botón *save image*. Ver figura 14.

Figura 14: pestaña *Analyze attributes* opción View Data.



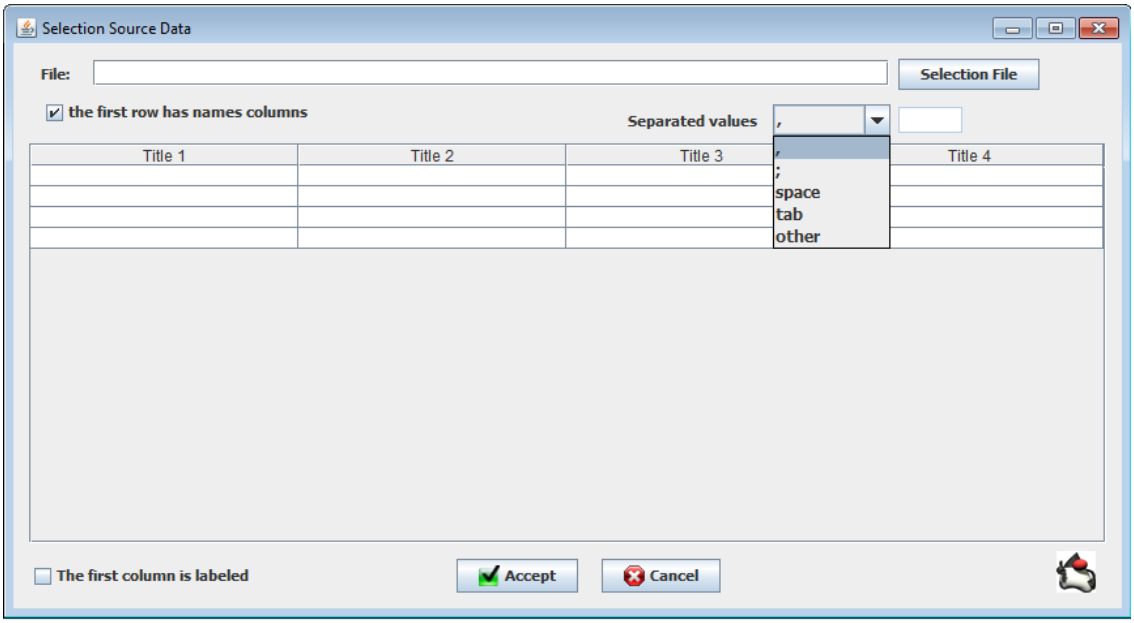
1.1.2 Icono Plain File

Permite la conexión a archivos separados por COMAS, PUNTO Y COMA, TAB, espacios u otros tipos de separadores. En la figura 15 se presentan las opciones de este icono.

Figura 15: Opciones Icono Plain File.

Configure
Despliega una ventana que permite seleccionar el archivo, presionando en el botón *Selection File*, en esta ventana también tiene la opción de establecer si se quiere tomar la primera fila como el nombre de la columnas seleccionando *the firts row has names columns*. Otra opción es seleccionar *the firts columns is labeled*, que permite tomar la primera columna como etiquetas o identificadores de los registros para su identificación. Si todos la configuración es correcta se habilita la opción *load*. También aquí se presenta una previsualización de los datos reconocidos.

Figura 16: Ventana de configuración Plain File.



Load
Permite cargar los datos del archivo a memoria. Sin esta opción no se puede conectar el icono con ningún otro icono.

View Data
Es igual a la opción del icono Data Base.

1.1.3 Icono Arff File

Permite la conexión a archivos con formato .ARFF que son archivos creados por la herramienta Weka. Las opciones *Configure, load y View Data* son iguales a las del Icono Plain File.

Figura 17: Opciones Icono Arff File.

1.1.4 Icono Excel File

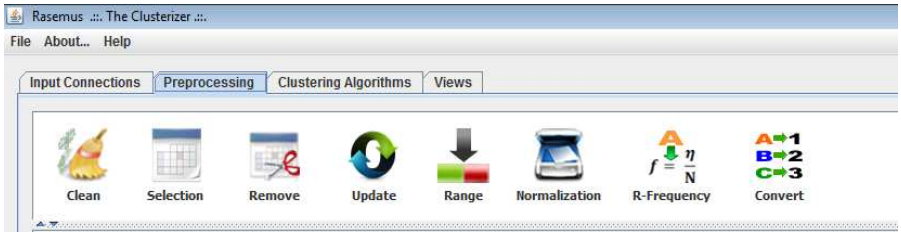
Permite la conexión a archivos con formato .XLS que son archivos creados, por Microsoft Excel. Las opciones *Configure, load y View Data* son iguales a las del Icono Plain File.

Figura 18: Opciones Icono Excel File.

1.2 *Preprocessing* o Preprocesamiento de Datos:

En la pestaña Input *Preprocesing*, se presentan los diferentes iconos que permiten limpieza, selección y transformación del conjunto de datos, entre ellos están el icono *Clean* que permite eliminar los datos de tipo nulo, el Icono *selection* , que permite seleccionar algunos atributos del conjunto de datos, el icono *Update* permite actualizar los valores de los atributos, el icono *Range* permite crear rangos en los atributos numéricos, el icono *Normalization* permite normalizar los atributos de tipo numérico, el icono *R-Frequency* reemplaza los atributos categóricos por el valor de la frecuencia relativa de la categoría dentro del atributo y por ultimo esta el icono *Convert* que convierte los atributos categóricos a numéricos asignándole un número determinado o también sirve para transformar atributos numéricos a categóricos. Estos se presentan en la figura 19.

Figura 19: iconos de la opción *Preprocesing.*



1.3 *Clustering Algorithms* o Algoritmos de Clustering

Permiten realizar las tareas de clustering a un conjunto de datos. Estos algoritmos se encuentran divididos en tres tipos, los de tipo de clustering particional o *Partitional Clustering* con los algoritmos kmeans, kprototype y kfrequency, los algoritmos de clustering jerárquicos o *Hierarchical Clustering* y los algoritmos de clustering basado en densidad o *Density Based Clustering*. Para poder ejecutar este tipo de opciones los iconos deben tener conectado un Icono Data Base, Plain File, Arff File, Excel File y todos los atributos de preprocesamiento. (ver figura 20)

Figura 20: iconos de la opción Clustering Algorithms.

1.3.1 Icono K-Means

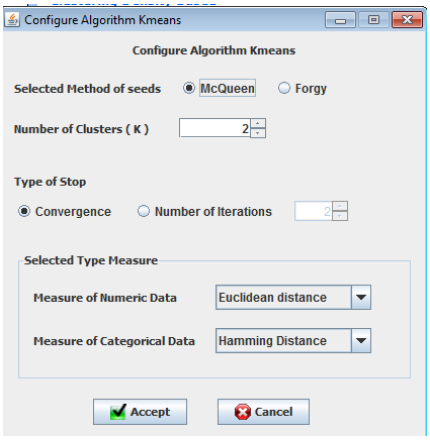
Crea segmentos del conjunto de datos a partir del algoritmo K-Means. En la figura 20 se muestras la opciones que presenta este icono (ver figura 21).

Figura 21: Opciones Icono K-Means.

Configure

Permite al usuario configurar el algoritmo kmeans, permitiéndole al usuario la opción de escoger el método de selección de semillas ya sea por McQueen que toma los primeros registros como semillas o la opción Forgy que toma semillas de forma aleatoria. La otra opción es definir el número de segmentos a formar. La siguiente es la opción de escoger el tipo de parada, entre las opciones están la convergencia, es decir cuando ya no haya cambios en los grupos la otra es según un determinado número de iteraciones. El siguiente parámetro permite elegir, qué formula de distancia se va a utilizar para los atributos numéricos y el otro que tipo de formula de similitud que se utilizara para los atributos categóricos, entre la formulas para tipos numéricos se tiene, la distancia euclidiana, la distancia Manhattan, distancia Chebyshev y mahalanobis y para los atributos categóricos se tiene las técnicas de similitud haming y proporción de coincidencias. Ver figura 22.

Figura 22: configuración icono kmeans



Run
Permite la ejecución de los algoritmos.

Add Attribute Cluster
Permite crear un archivo con formato .ARFF adicionando una columna en donde ira el numero del segmento al cual fue asignado el registro.

1.3.2 Icono K-Prototype

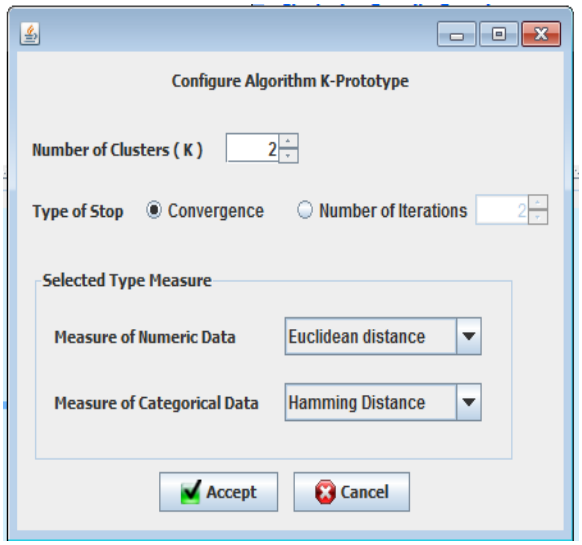
Crea segmentos del conjunto de datos a partir del algoritmo K-Prototype. En la figura 23 se muestran las opciones que presenta este icono.

Figura 23: Opciones icono kprototype.

Configure

Permite al usuario configurar el algoritmo kprototype, permitiéndole al usuario la opción de definir el número de segmentos a formar. La siguiente es la opción de escoger el tipo de parada, entre las opciones están la convergencia, es decir cuando ya no haya cambios en los grupos la otra es según un determinado número de iteraciones. El siguiente parámetro permite elegir, qué formula de distancia se va a utilizar para los atributos numéricos y el otro que tipo de formula de similitud que se utilizara para los atributos categóricos, entre la formulas para tipos numéricos se tiene, la distancia euclidiana, la distancia Manhattan, distancia Chebyshev y mahalanobis y para los atributos categóricos se tiene las técnicas de similitud haming y proporción de coincidencias. Ver figura 24.

Figura 24: Configuración icono K-Prototype



Run

Permite la ejecución de los algoritmos.

Add Attribute Cluster

Permite crear un archivo con formato .ARFF adicionando una columna en donde ira el numero del segmento al cual fue asignado el registro.

1.3.3 Icono wayCluster

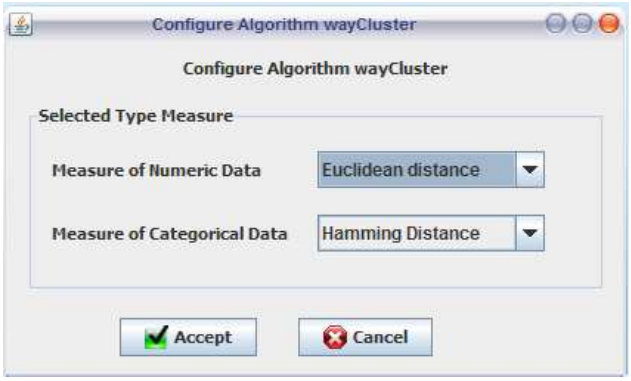
Crea segmentos del conjunto de datos a partir del algoritmo wayCluster. En la figura 25 se muestran las opciones que presenta este icono.

Figura 25: Opciones icono wayCluster.

Configure

Permite al usuario configurar el algoritmo wayCluster. Solo permite elegir qué formula de distancia se va a utilizar para los atributos numéricos y el otro que tipo de formula de similitud se utilizara para los atributos categóricos, entre la formulas para tipos numéricos se tiene, la distancia euclidiana, la distancia Manhattan, distancia Chebyshev y mahalanobis y para los atributos categóricos se tiene las técnicas de similitud haming y proporción de coincidencias. (ver figura 26)

Figura 26: Configuración icono wayCluster



Run
Permite la ejecución de los algoritmos.

Add Attribute Cluster
Permite crear un archivo con formato .ARFF adicionando una columna en donde ira el numero del segmento al cual fue asignado el registro.

1.3.4 Icono K-Frequency

Crea segmentos del conjunto de datos a partir del algoritmo K-Frequency. En la figura 27 se muestran las opciones que presenta este icono.

Figura 27: Opciones icono K-Frequency

Configure

Permite al usuario configurar el algoritmo kfrequency, permitiéndole al usuario la opción de escoger el método de selección de semillas, utilizando los primeros registros como semillas utilizando la opción aleatoria que toma semillas aleatorias. La otra opción es definir el número de segmentos a formar. La siguiente es la opción de escoger el tipo de parada, entre las opciones están la convergencia, es decir cuando ya no haya cambios en los grupos o según un determinado número de iteraciones, el siguiente parámetro permite elegir qué formula de distancia se va a utilizar para los atributos numéricos entre las formulas se tiene la distancia euclidiana, la distancia Manhattan, distancia Chebyshev y mahalanobis y para los atributos categóricos se tiene las técnicas de similitud haming y proporción de coincidencias. Ver figura 28.

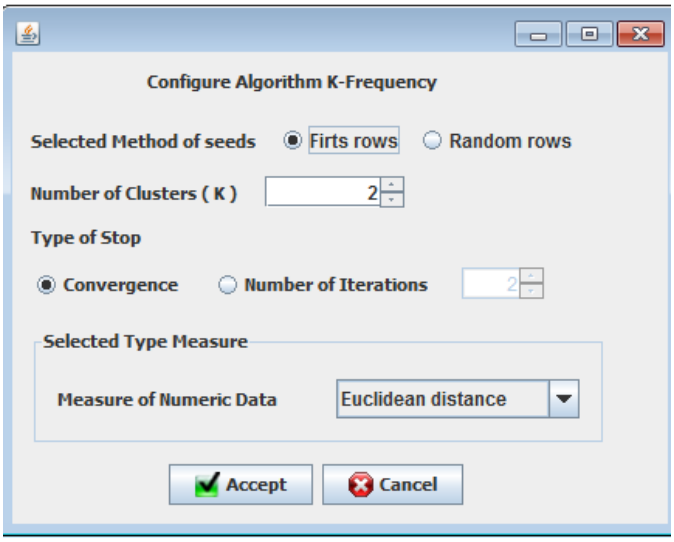
Run

Permite la ejecución de los algoritmos.

Add Attribute Cluster

Permite crear un archivo con formato .ARFF adicionando una columna en donde ira el numero del segmento al cual fue asignado el registro.

Figura 28: configuración icono K-Frequency.



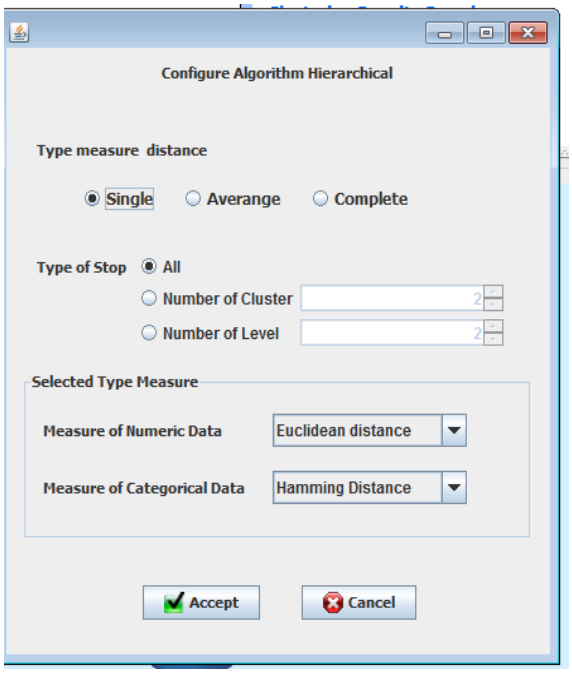
1.3.5 Icono Hierarchical

Crea segmentos del conjunto de datos a partir de algoritmos Jerárquicos de tipo aglomerativo. En la figura 29 se muestran la opciones que presenta este icono.

Figura 29: Opciones icono Hierarchical

Configure
Permite al usuario configurar que tipo de algoritmo jerárquico se desea utilizar, estos son: el algoritmo jerárquico de enlace mínimo, el cual corresponde a la opción *Single*, seguido de la opción de clustering jerárquico de enlace promedio que corresponde a la opción *Averange* y por último se tiene la opción complete que permite ejecutar el algoritmo con enlace máximo. La siguiente es la opción de escoger el tipo de parada, entre las opciones está formar un solo segmento o formar un determinado número de segmentos o alcanzar un cierto número de niveles o iteraciones en la creación del dendograma. , el siguiente parámetro permite elegir qué formula de distancia se va a utilizar para los atributos numéricos y el otro que tipo de formula de similitud se utilizara para los atributos categóricos, entre la formulas para tipos numéricos se tiene, la distancia euclidiana, la distancia Manhattan, distancia Chebyshev y mahalanobis y para los atributos categóricos se tiene las técnicas de similitud haming y proporción de coincidencias. Ver figura 30.

Figura 30: Configuración icono Hierarchical.



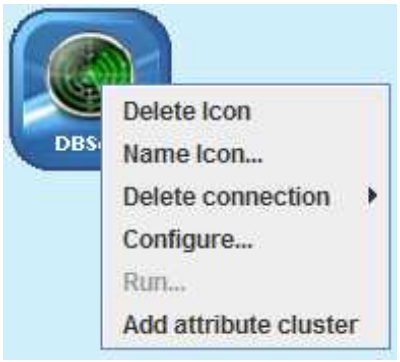
Run
Permite la ejecución de los algoritmos.

Add Attribute Cluster
Permite crear un archivo con formato .ARFF adicionando una columna en donde ira el numero del segmento al cual fue asignado el registro.

1.3.5 Icono DBScan

Crea segmentos del conjunto de datos a partir del algoritmo DBScan, que hace parte de los algoritmos de clustering basado en densidad. En la figura 31 se muestran la opciones que presenta este icono.

Figura 31: Opciones icono Hierarchical



Configure

Permite al usuario configurar los parámetros del algoritmo DBScan, estos son: el tamaño del área de barrido o el grado de densidad denominado épsilon. El siguiente parámetro es el número mínimo de puntos que debe haber en área de barrido para considerarse segmento. El tipo d barrido puede definir si va a ser solo el barrido directo o *Directly Density – Reachable* o la opción de barrido conectado o *Density - Connected* que permite la unión de elementos que estén cerca de los elementos segmentados. El siguiente parámetro permite elegir qué formula de distancia se va a utilizar para los atributos numéricos y el otro que tipo de formula de similitud se utilizara para los atributos categóricos, entre la formulas para tipos numéricos se tiene, la distancia euclidiana, la distancia Manhattan, distancia Chebyshev y mahalanobis y para los atributos categóricos se tiene las técnicas de similitud haming y proporción de coincidencias. Ver figura 32.

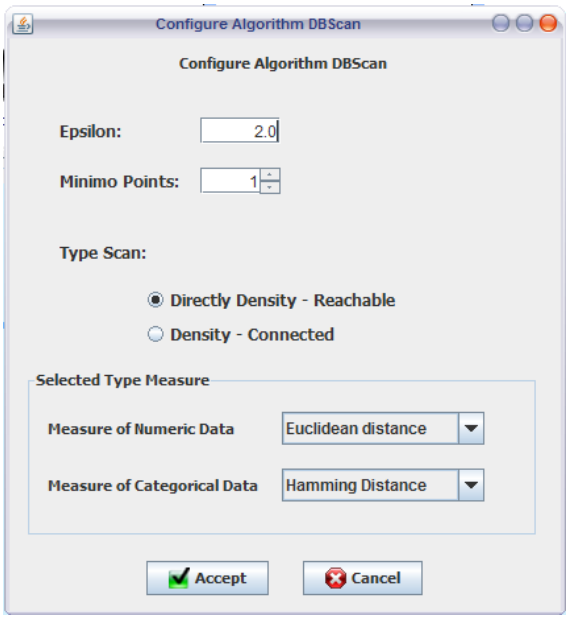
Run

Permite la ejecución de los algoritmos.

Add Attribute Cluster

Permite crear un archivo con formato .ARFF adicionando una columna en donde ira el numero del segmento al cual fue asignado el registro.

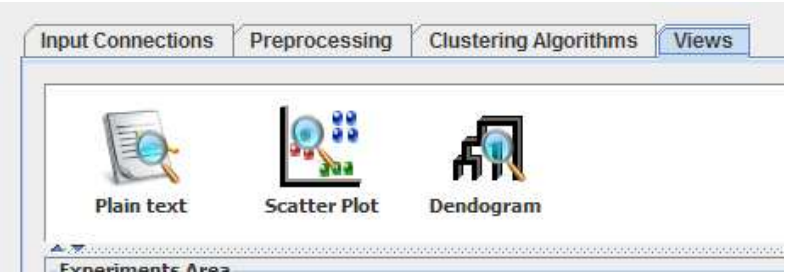
Figura 32: configuración icono DBScan.



1.4 Views o Visualización de Resultados

Permite al usuario ver los resultados de los diferentes algoritmos de clustering aplicados. Se manejan tres tipos de opciones que permiten analizar los resultados desde diferentes puntos de vista. Son fácil manejo y permiten evaluar la calidad de los segmentos formados. Entre ellos se tiene la opcion *Plan text*, que despliega los resultados en formato plano y tablas, la opcion *Scatter Plot* que le presenta al usuario los diferentes tipo de diagras de dispersion y por ultimo que es esclusivo de los algoritmos jerarquicos que la presentacion del diagrama resultado. Ver figura 33.

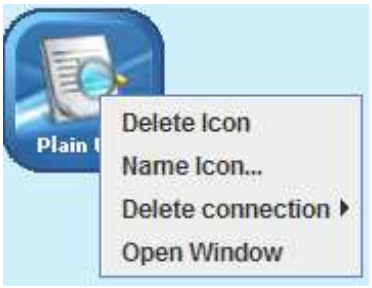
Figura 33: opciones de vsualización de resultados.



1.4.1 icono Plain text

Permite ver un informe de resultados de la ejecución de un algoritmo de clustering. Sus opciones se presentan en la figura 34.

Figura 34: Opciones icono Plain Text.



Open Window

Permite al usuario desplegar una ventana, que está compuesta por diferentes pestañas que permiten hacer un análisis diferente de los datos. La primera pestaña es *Plain Text* que presenta un informe en donde se tienen aspecto como el error cuadrático, los representantes de clúster, la desviación estándar de los segmentos formados y la distancia euclidiana entre los representantes de clúster. Este informe se puede exportar a archivos con formato .PDF, .RFT y HTML (ver figura 35).

La siguiente pestaña se denomina *Data Assigment*, que permite ver en que segmento fue asignado determinado registro (ver figura 36).

Otra opción es la pestaña *Representatives of Cluster* que presenta los representantes de los segmentos formados.(ver figura 37)

Figura 35: Imagen de la opción del icono Plain Text , pestaña *Plain Text*

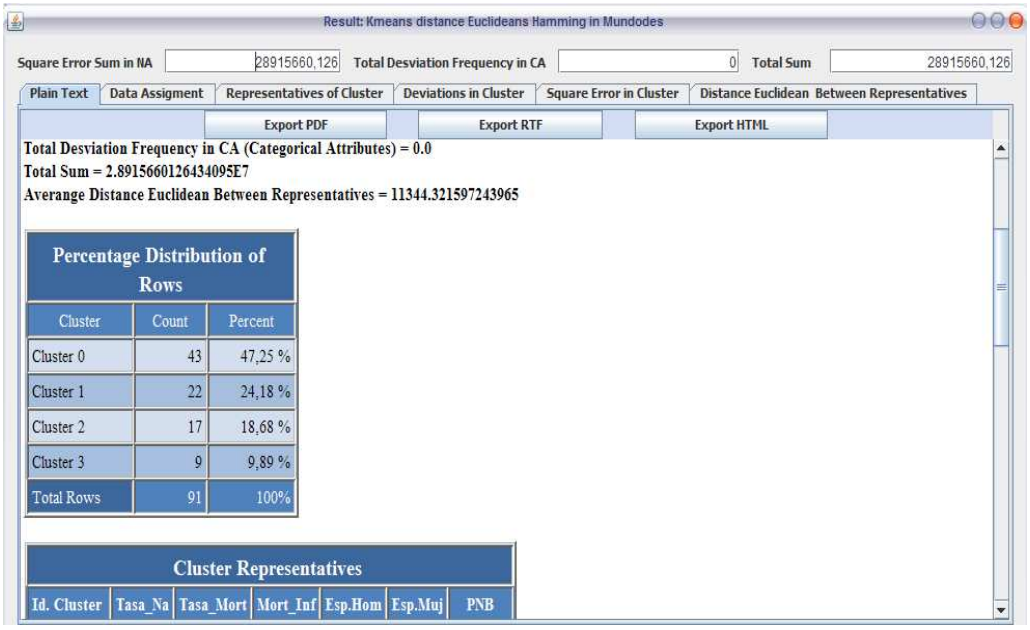
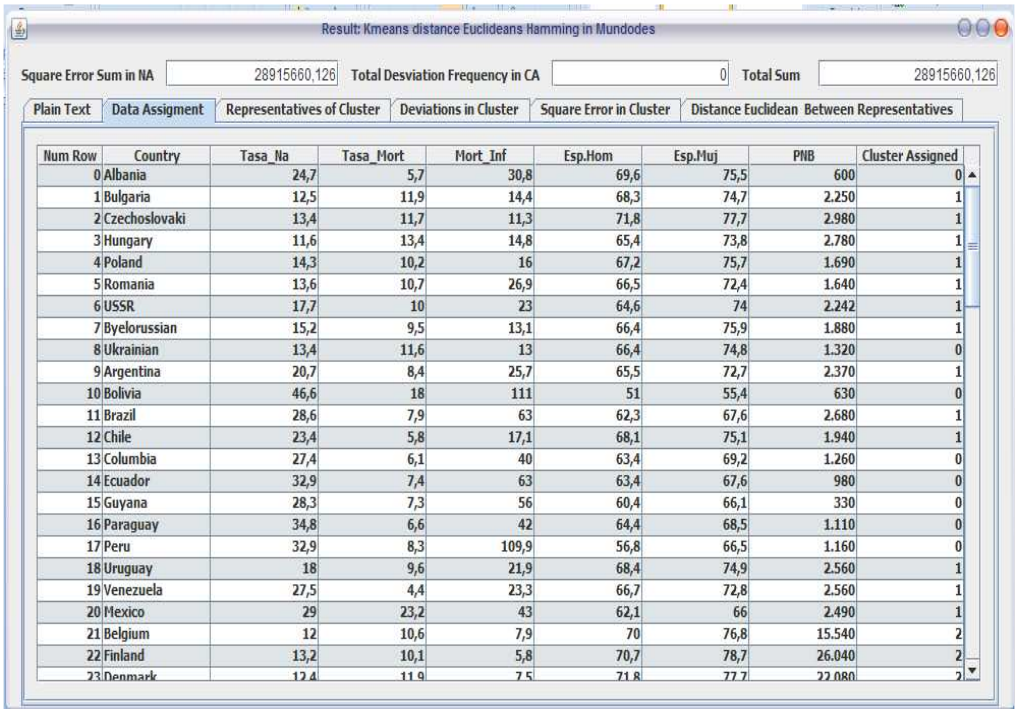
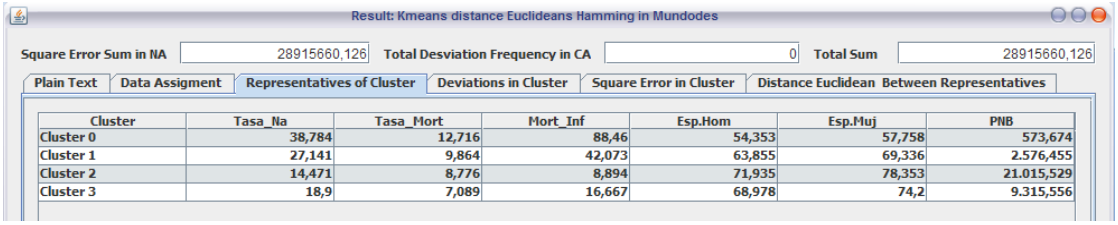


Figura 36: Imagen de la opción del icono Plain Text , pestaña *Data Assigment*



Num Row	Country	Tasa_Na	Tasa_Mort	Mort_Inf	Esp.Hom	Esp.Muj	PNB	Cluster Assigned
0	Albania	24,7	5,7	30,8	69,6	75,5	600	0
1	Bulgaria	12,5	11,9	14,4	68,3	74,7	2.250	1
2	Czechoslovaki	13,4	11,7	11,3	71,8	77,7	2.980	1
3	Hungary	11,6	13,4	14,8	65,4	73,8	2.780	1
4	Poland	14,3	10,2	16	67,2	75,7	1.690	1
5	Romania	13,6	10,7	26,9	66,5	72,4	1.640	1
6	USSR	17,7	10	23	64,6	74	2.242	1
7	Byelorussian	15,2	9,5	13,1	66,4	75,9	1.880	1
8	Ukrainian	13,4	11,6	13	66,4	74,8	1.320	0
9	Argentina	20,7	8,4	25,7	65,5	72,7	2.370	1
10	Bolivia	46,6	18	111	51	55,4	630	0
11	Brazil	28,6	7,9	63	62,3	67,6	2.680	1
12	Chile	23,4	5,8	17,1	68,1	75,1	1.940	1
13	Columbia	27,4	6,1	40	63,4	69,2	1.260	0
14	Ecuador	32,9	7,4	63	63,4	67,6	980	0
15	Guyana	28,3	7,3	56	60,4	66,1	330	0
16	Paraguay	34,8	6,6	42	64,4	68,5	1.110	0
17	Peru	32,9	8,3	109,9	56,8	66,5	1.160	0
18	Uruguay	18	9,6	21,9	68,4	74,9	2.560	1
19	Venezuela	27,5	4,4	23,3	66,7	72,8	2.560	1
20	Mexico	29	23,2	43	62,1	66	2.490	1
21	Belgium	12	10,6	7,9	70	76,8	15.540	2
22	Finland	13,2	10,1	5,8	70,7	78,7	26.040	2
23	Denmark	12,4	11,9	7,5	71,8	77,7	22.080	2

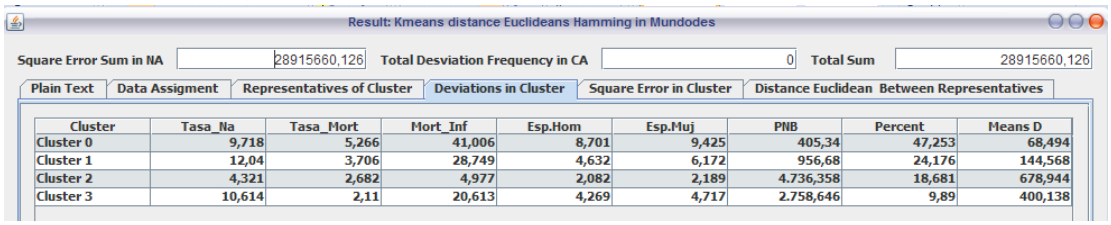
Figura 37: Imagen de la opción del icono Plain Text , pestaña *Representatives of Cluster.*



Cluster	Tasa_Na	Tasa_Mort	Mort_Inf	Esp.Hom	Esp.Muj	PNB
Cluster 0	38,784	12,716	88,46	54,353	57,758	573,674
Cluster 1	27,141	9,864	42,073	63,855	69,336	2.576,455
Cluster 2	14,471	8,776	8,894	71,935	78,353	21.015,529
Cluster 3	18,9	7,089	16,667	68,978	74,2	9.315,556

Otra opción es la pestaña *Deviations in Cluster* que presenta las desviaciones estándar para los atributos numéricos y el error de frecuencia de los atributos categóricos.(ver figura 37)

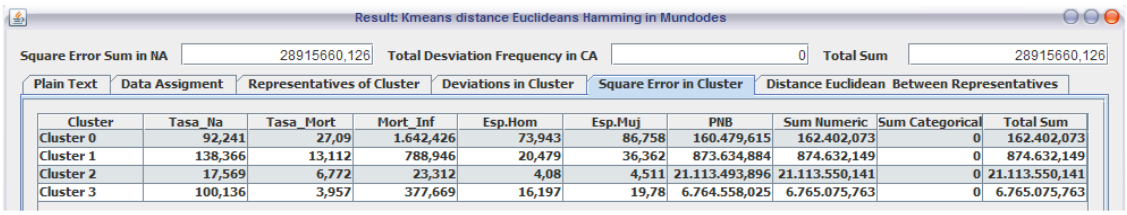
Figura 38: Imagen de la opción del icono Plain Text , pestaña *Deviations in Cluster.*



Cluster	Tasa_Na	Tasa_Mort	Mort_Inf	Esp.Hom	Esp.Muj	PNB	Percent	Means D
Cluster 0	9,718	5,266	41,006	8,701	9,425	405,34	47,253	68,494
Cluster 1	12,04	3,706	28,749	4,632	6,172	956,68	24,176	144,568
Cluster 2	4,321	2,682	4,977	2,082	2,189	4.736,358	18,681	678,944
Cluster 3	10,614	2,11	20,613	4,269	4,717	2.758,646	9,89	400,138

Otra opción es la pestaña *Square Error in Cluster* que presenta los errores cuadráticos promedio de los atributos de tipo numérico y el error de frecuencia de los atributos categóricos.(ver figura 39)

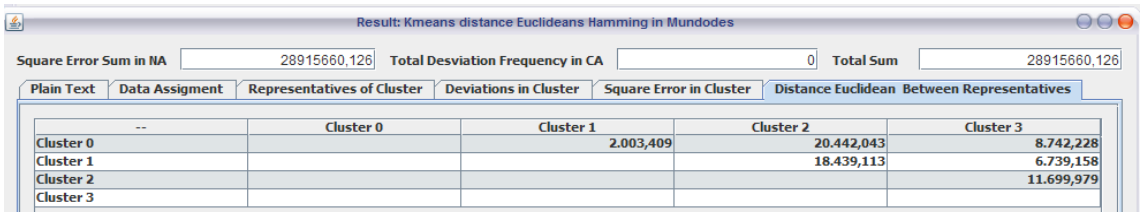
Figura 39: Imagen de la opción del icono Plain Text , pestaña *Square Error in Cluster.*



Cluster	Tasa_Na	Tasa_Mort	Mort_Inf	Esp.Hom	Esp.Muj	PNB	Sum Numeric	Sum Categorical	Total Sum
Cluster 0	92,241	27,09	1.642,426	73,943	86,758	160.479,615	162.402,073	0	162.402,073
Cluster 1	138,366	13,112	788,946	20,479	36,362	873.634,884	874.632,149	0	874.632,149
Cluster 2	17,569	6,772	23,312	4,08	4,511	21.113.493,896	21.113.550,141	0	21.113.550,141
Cluster 3	100,136	3,957	377,669	16,197	19,78	6.764.558,025	6.765.075,763	0	6.765.075,763

Y la ultima pestaña es *Distance Euclidean Between Representatives* que presenta la distancia euclidiana para los numéricos y haming para los categóricos, entre los representantes de grupo. (ver figura 40)

Figura 40: Imagen de la opción del icono Plain Text , pestaña *Distance Euclidean Between Representatives.*



--	Cluster 0	Cluster 1	Cluster 2	Cluster 3
Cluster 0		2.003,409	20.442,043	8.742,228
Cluster 1			18.439,113	6.739,158
Cluster 2				11.699,979
Cluster 3				

1.4.1 Icono Scatter Plot

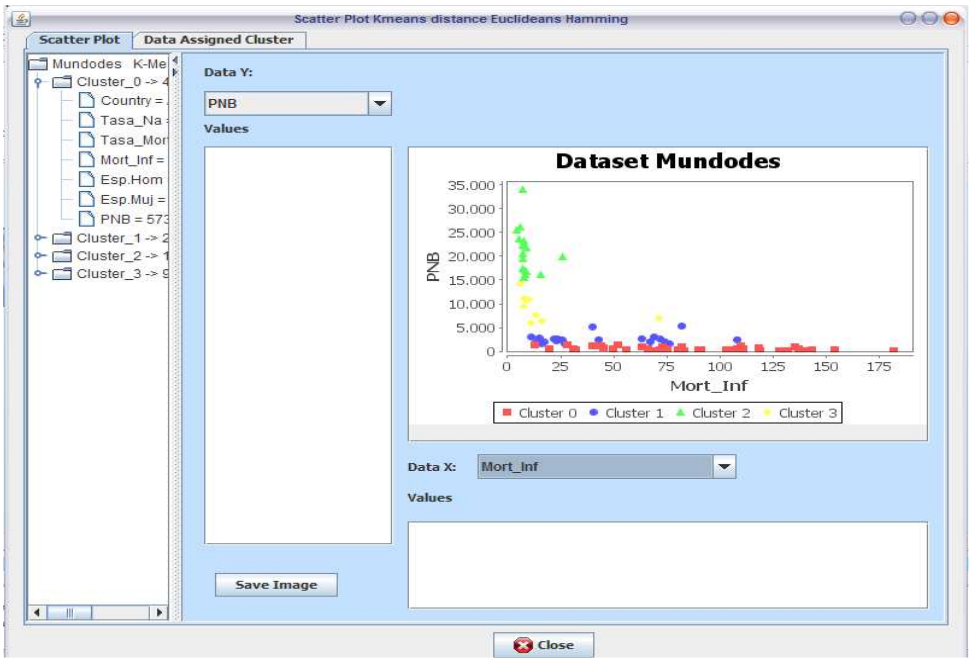
Permite ver un diagrama de dispersión X vs Y de los atributos, diferenciando la asignación de los segmentos formados. Sus opciones se presentan en la figura 41.

Figura 41: Opciones icono Scatter Plot.

Open Window

Despliega una ventana que le permite interactuar al usuario en la creación de diagramas de dispersión X vs Y, brindándole la posibilidad de guardar las imágenes en formato .JPG.

Figura 42: Visualización de Diagramas de dispersión.



1.4.1 Icono Dendogram

Permite ver el dendograma formado por los algoritmos de clustering jerárquico, Sus opciones se presentan en la figura 43.

Figura 43: Opciones icono Scatter Plot.

Open Window

Permite al usuario analizar la jerarquía generada en los algoritmos de clustering jerárquico diferenciando segmentos formados. Si el usuario al ejecutar los

algoritmos jerárquicos coloco un numero de segmentos a formar, estos se presentaran con diferentes colores, además de hacer un efecto de zoom para mejor comprensión del grafico con el Scroll del Mouse, también le da la posibilidad de guardar las imágenes en formato .JPG. Ver figura 44.

Figura 44: Visualización del Dendograma.