UNIVERSIDAD POLITÉCNICA DE VALENCIA

Análisis comparativo DBDT vs otros Algoritmos para el manejo de datos no escalares.

Tesis de Máster

José Luis Ramírez Cabrera

Departamento de Sistemas Informáticos y Computación.
Universidad Politécnica de Valencia

Septiembre de 2013

Análisis comparativo DBDT vs otros algoritmos para el manejo de datos no escalares.

Tesis de Máster

Máster en Ingeniería del Software, Métodos Formales y Sistemas de Información

Dirigida por el Doctor:

José Hernández Orallo

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Septiembre 2013

A mi Familia, por Su Solidaridad y Amor
A mis Profesores, por orientar mi camino.
A quienes me acompañaron, pues lograron hacer mejor

Índice

1. Introducción

1.1 Motivación

La ciencia de extraer información útil de grandes conjunto de datos o bases de datos es conocida como minería de datos (1). En esta disciplina, relativamente nueva convergen diferentes disciplinas como la estadística, el aprendizaje automático (Machine Learning), la administración de base de datos, la inteligencia artificial, la estadística, la recuperación de información entre otras. Dentro de estas áreas, es el aprendizaje automático el que se ocupa de extraer patrones a partir de los datos, donde usando varias técnicas buscan predecir o describir datos.

En los últimos años, con el uso del aprendizaje automático, se ha logrado cambiar el paradigma de cómo se entienden, interpretan y utilizan los datos. Este afán de explotar la información ha logrado diferentes mejoras y ampliaciones en algoritmos existentes (2 pág. 39), arquitecturas integradas que permiten combinar procedimientos de búsqueda, inferencia y representación o sus aplicaciones en el descubrimiento de conocimiento en internet en campos como : filtrado de información (3), la búsqueda inteligente (4), la traducción automática (5), o como se ha mencionado antes la minería de datos (6).

Para realizar las tareas de clasificación, se utilizan diferentes algoritmos como los vecinos más próximos (KNN), los árboles de decisión (ID3, C.4.5, CART), las redes neuronales, los métodos probabilísticos que han demostrado su efectividad. Tradicionalmente, las tareas del aprendizaje automático se han concentrado en tipos de datos numéricos o nominales, la necesidad de explorar los diferentes tipos de información, ha despertado la curiosidad de explorar los datos con estructuras complejas como secuencias, grafos... etc. Los datos con estructuras complejas tienden a ser usados como un tipo de dato nominal al que se le asigna un valor discreto, perdiendo así el valor intrínseco del tipo de dato. Por ejemplo, si se trata de comparar dos moléculas como el agua (H20) y el sulfuro de hidrogeno (H2S), intuitivamente sabemos que son parecidas, pero si se quiere ser exacto es necesario introducir algún método que permita compararlas. Calcular la distancia, es una manera de abordar el problema de datos con estructuras complejas. El DBDT(Distance Based Decisión Tree) (7 pág. 198), es un nuevo algoritmo que permite extender el uso de las técnicas de clasificaciones actuales a datos con estructuras complejas como conjuntos, listas, conjuntos de gráficos múltiples, y datos estructurados, entre otras. Es la motivación de este trabajo, explorar datos a través del uso de una nueva técnica conocida como DBDT (Distance Based Decisión Tree) (7 pág. 198), y comparar su resultado respecto a las técnicas tradicionales.

1.2 Objetivos

En el presente trabajo, se pretende experimentar con un nuevo método basado en Distancias, conocido como DBDT (Distance Based Decisión Tree), para evaluar el aporte al campo de la experimentación en la minería de datos y el aprendizaje automático de datos no escalares. Teniendo en cuenta lo anterior, se han trazado como objetivos para este trabajo, los descritos a continuación:

- Implementar la distancia de edición (Levenshtein) para con el fin de ser usada con el DBDT.
- Realizar pruebas con el DBDT sobre datos que contengan atributos que sean cadenas.
- Comparar de acuerdo a distintas medidas de evaluación, el DBDT respecto a otros algoritmos.
- Determinar las perspectivas del DBDT para este tipo de problemas.

2 Minería de Datos y Aprendizaje Automático

La necesidad de manejar grandes cantidad de datos ha sido uno de los grandes retos de los últimos tiempos. Como respuesta a este ha surgido como disciplina la minería de datos y el aprendizaje automático. Las tareas propias de la minería de datos y el aprendizaje automático, pueden ser clasificadas en dos tipos: clasificación y predicción, estas tareas han cobrado importancia en los últimos años debido a al auge de la informática en el mundo moderno y la capacidad de recolección de datos de los sistemas de información vigentes. A continuación se describen y definen estas técnicas.

2.1 Minería de Datos y Aprendizaje automático

Para (8), la minería de datos es *"el proceso de extraer conocimiento útil y comprensible, previamente desconocido, desde grandes cantidades de datos almacenados en distintos formatos".*

Otros como (1)definen la minería de datos como *"el análisis de conjuntos de datos observacionales para encontrar relaciones insospechadas y sumariar los datos de una manera novedosa que sea entendible y útil para el propietario de los datos".*

La minería de datos es una disciplina donde convergen diferentes áreas del conocimiento (ver Ilustración 1). Esta convergencia, hace difícil definir los límites estrictos en cada una de las disciplinas. Los límites dependen de la tarea, los datos o las necesidades de los usuarios. El proceso que lleva a esta disciplina a buscar relaciones en un conjunto de datos, se desarrolla a través de un número de pasos:

1. Determinar la naturaleza y la estructura de la representación de los datos para ser usada.
2. Procesar los datos (transformar, seleccionar y limpiar) y determinar la tarea a realizar.
3. Escoger los algoritmos para procesar y determinar la función de evaluación.
4. Decidir que principios del manejo de datos, son requeridos para implementar los diferentes algoritmos.

Una vez existen los datos es indispensable saber que tareas se puede realizar sobre estos. Algunas de las tareas de la minería de datos propuestas por (9) son:

- ***Análisis de datos exploratorio:*** Como lo indica su nombre, tiene como objetivo explorar los datos sin tener ideas claras acerca del tipo de conocimiento que se pretende encontrar. Estas técnicas son interactivas y visuales.

Ilustración 1 Disciplinas que contribuyen a la minería de datos

- **Modelamiento descriptivo:** Este permite describir o explicar patrones en los datos. Los ejemplos de estas descripciones incluyen modelos para determinar la distribución de probabilidad de los datos (*estimación de densidad*), aprisionamiento del espacio dimensional en grupos (*clúster y análisis de segmentación*) y modelos que describen relaciones sobre las variables (*modelamiento de dependencia*). Un ejemplo de esta técnica es al análisis de segmentación, donde se dividen los datos en grupos homogéneos.

- ***Modelamiento predictivo: clasificación y regresión.*** Estas tareas tienen como objetivo construir un modelo que permite predecir una variable de valores conocidos de otras variables. Para problemas de clasificación, la variable que se busca predecir es de tipo categórico, mientras en los problemas de regresión la variable es cuantitativa. La principal diferencia entre predicción y descripción, es que en la predicción tiene como único objetivo predecir el valor de una variable, mientras en los problemas descriptivos no están centrados en una variable. Un ejemplo clásico de esto es es el de predecir el valor de una acción de un mercado.

- ***Descubrimiento de subgrupos:*** Las tareas anteriores buscan construir modelos, esta busca encontrar patrones .Esos patrones pueden ser un comportamiento específico, por lo cual se encarga de buscar o definir comportamientos basados en una diferencia significativa de los datos. Un ejemplo del uso de esta técnica son los algoritmos de reglas de asociación.

- ***Recuperación de contenido:*** Esta tarea de la minería de datos permite a través un patrón, encontrar patrones similares en un conjunto de datos aunque a veces se suelen incluir dentro del modelamiento descriptivo. Esta tarea es aplicada de manera común y para los conjuntos de datos que contienen textos e imágenes, está más relacionada con la recuperación de información.

Muchas de las tareas descritas anteriormente son muy diferentes pero sus algoritmos mantienen componentes comunes. Estos componentes según (1) son:

- Estructura del modelo o patrón: determinan la estructura subyacente o funcional.

- Función de puntuación: tiene como objetivo juzgar la calidad de un modelo entrenado.

- Método de optimización y búsqueda: optimiza la función de puntuación y búsqueda sobre diferentes estructuras de modelos o patrones.

- Estrategia de manejo de datos: el manejo del acceso a datos de manera eficiente.

2.2 Aprendizaje automático

Como se ha descrito anteriormente, una de las disciplinas de la minería de datos es el aprendizaje automático. Esta se encarga de llevar las técnicas del aprendizaje humano a lenguaje de computadora. Su objetivo es inducir al conocimiento partiendo de un conocimiento previo para poder seguir aprendiendo una vez que este conocimiento se va actualizando. De acuerdo a lo aprendido, el ordenador crea un modelo (clasificación u optimización) que a partir de la retro alimentación, mejora su desempeño. Según el tipo de retroalimentación se puede conocer la naturaleza del paradigma de aprendizaje que va resolver el modelo. Según (10), se distinguen tres tipos de paradigmas de aprendizaje:

- Aprendizaje supervisado, el modelo aprende a partir de ejemplos de sus entradas y salidas. Su objetivo es predecir un determinado valor a partir de una serie de ejemplos dados previamente (datos de entrenamiento). Si la realimentación proporciona un valor correcto para los ejemplos, se puede decir que es aprendizaje supervisado.

- Aprendizaje no supervisado, busca que el modelo aprenda a partir de sus entradas sin conocer el valor de sus salidas. Es decir el modelo debe proveer información, donde sepa diferenciar que información es correcta o incorrecta sin tener información previa sobre lo que es correcto o no.

- Aprendizaje por refuerzo: Este es el más general de los tres. En este caso el modelo puede aprender que hacer, cuando no existe quien le enseñe que acción tomar en cada situación. Aquí, la realimentación indica que es bueno y que es malo, para así saber qué hacer en la siguiente situación.

En el aprendizaje automático existen diferentes técnicas para realizar las tareas de clasificación o predicción, siendo dos en particular las utilizadas para esta tesis:1) métodos basados en casos y 2) los arboles de decisión, con el fin de entender mejor esto métodos son expuestos a continuación

2.3 Métodos basados en distancia

Los métodos basados en distancias, parten del razonamiento por analogía, donde si se tienen problemas similares se pueden aplicar soluciones similares, son un caso particular del razonamiento basado en casos (CBR), del inglés "case-based-reasoning". Usualmente los métodos basados en distancia, se clasifican de acuerdo al momento en que se utiliza la distancia: si la distancia es utilizada de manera anticipada, se utiliza el modelo para generalizar todos los ejemplos a futuro; se puede generalizar de forma retardada, donde se espera a tener el nuevo ejemplo para generalizar. Para poder construir los modelos es importante entender cómo funciona la distancia y como se trata la distancia dependiendo del tipo de objetos (atributos), como se expone a continuación.

2.3.1 Distancias

Los métodos basados en distancia utilizan definiciones clásicas de distancia para realizar las diferentes comparaciones entre objetos. Varias de las técnicas de Minería de Datos por ejemplo KNN, Análisis de clúster o métodos de escalado multidimensional, buscan obtener medidas similares entre objetos usando distancias.

Para introducir el término de distancia es preciso hablar de la disimilitud entre objetos y cómo se puede medir la misma. Al referirse a distancia se puede tomar la definición de (1 pág. 33), donde se define distancia como una medida de disimilitud, que satisface las siguientes tres condiciones:

1. $d(i,j) \geq 0 \; \forall \; i,j \;\; \wedge \;\; d(i,j) \;\; = 0 \; sí \; y \; solo \; sí \; i = j$
2. $d(i,j) = d(j,i) \; \forall \; i \; , j$
3. $d(i,j) \leq d(i,k) + d(k,j) \;\; \forall \;\; i,j,k \; (\; desigualdad \; triangular)$

La disimilitud, presenta una forma idónea de medir la diferencia entre objetos, a partir de la distancia y característica de los mismos. Algunas de las distancias más usadas se describen formalmente en los diferentes algoritmos tal como lo presenta (1).

La distancia euclidiana es una de las distancia más conocidas y usadas .Si se tiene un conjunto de observaciones $x(i) = \left(x_1(i), x_2(i) \dots, x_p(i)\right)$, $1 \leq i \leq n$, donde el valor de la k-ésima variable para el i – ésimo elementos es $X_k(i)$. Entonces, la distancia euclidiana entre i y j elementos se puede definir como:

$$d(i,j) = \sqrt[2]{\sum_{k=1}^{p} \left(x_k(i) - x_k(j)\right)^2}$$

Derivada de la distancia euclidiana encontramos también la distancia euclidiana ponderada, definida como:

$$d(i,j) = \sqrt[2]{\sum_{k=1}^{p} W_k\left(x_k(i) - x_k(j)\right)^2}$$

La distancia euclidiana ponderada permite en los conjuntos de datos normalizar la contribución de una variable, independiente de su unidad.

Otra distancia muy utilizada es la distancia de Minkowsky o $\mathcal{L}\, p$, que se define como:

$$d(i,j) = \sqrt[q]{\sum_{K=1}^{q} |(x_k(i) - x_k(j))|^q}\,,\ donde\ q \geq 1$$

Esta distancia en particular tiene dos casos espaciales. Si q=2, es un caso especial de la distancia euclidiana. Si q=1 representa la distancia de Manhattan. Otro tipo de distancia usada comúnmente es la distancia de Mahalanobis, definida como:

$$d(i,j) = \sqrt[2]{\left(x(i) - x(j)\right)^t S^{-1} (x(i) - x(j))}$$

Donde S representa la matriz de covarianza, que estandariza los datos en un espacio dimensional.

Las distancias descritas suelen utilizarse para tipos de atributos numéricos, pero los conjuntos de datos usualmente suelen tener más de un tipo de atributo como atributos nominales y datos estructurados (secuencias, listas., etc….). Es por esto que se han definido también distancias para estos tipos de datos .

Para manejar la distancia en atributos nominales, es común utilizar la siguiente función:

$$d(i,j) = \omega \sum_{i=1}^{n} \delta(i,j)$$

Donde ω es un factor de reducción y δ es una función donde $\delta(i,j) = 0$ si y solo si $i = j$ y $\delta(i,j) = 1$ en caso contrario. En el caso de los datos estructurados como secuencias, se suele usar una distancia conocida como Distancia de Levenshtein o distancia de edición, esta distancia, cuenta con número mínimo de operaciones (inserciones, borrados y sustituciones) que son necesarias para transformar una secuencia en otra. Es decir, la distancia correspondiente es el valor mínimo requerido para realizar la transformación de una secuencia A, en una secuencia B. ¿Cómo se calcula la distancia de edición? A continuación, un ejemplo. Se tienen dos cadenas s1 y s2, Si definimos s_1 = "casal" y s_2 = "carta", la transformación óptima (la más eficiente) para transformar s_1 en s_2 consiste en la sustitución del símbolo "s" por el símbolo "r", seguido por la adición de un símbolo "t" de s_1 y luego la eliminación del símbolo "l" de s_1 Como se muestra a continuación:

$$\begin{matrix} casal \\ carta \end{matrix}$$

$$s_1 = casal \rightarrow caral \rightarrow cartal \rightarrow carta = s_2;$$

Todo esto hace que d (s_1, s_2) = 3, para la transformación óptima. Es preciso aclarar, que no hay una única transformación óptima, puede existir más de una transformación óptima. Estas son algunas de las distancias más comunes usadas en las tareas propias de la minería de datos, con el fin de profundizar en el conocimiento de estas técnicas a continuación se hace una introducción de estas en el siguiente numeral.

2.4 Árboles de decisión

Los árboles de decisión hacen parte de los modelos comprensibles, estos modelos como lo indica su nombre son fácilmente interpretados por los usuarios, pues permiten visualizar fácilmente, el resultado del uso de esta técnica. Los arboles de decisión permiten establecer un modelo a partir de un conjunto de datos de entrenamiento, que puede ser representado de manera simbólica. Esto permite establecer un conjunto de condiciones que pueden ser interpretables fácilmente por los seres humanos

Una definición de árbol de decisión se puede tomar de (11):

"Un árbol de decisión es un conjunto de condiciones organizadas en una estructura jerárquica, de tal manera que la decisión final a tomar se puede determinar siguiendo las condiciones que se cumplen desde la raíz del árbol hasta alguna de las hojas"

Los árboles de decisión, realizan una clasificación de instancias partiendo del nodo raíz hasta llegar a alguna rama de un nodo hijo. Cada uno de los nodos hijo, es la representación de un atributo y cada rama representa un posible valor de un atributo. Este mismo proceso se realiza de manera recursiva para cada subárbol, que se agrega a cada nodo como se puede ver en la Ilustración 2.

El algoritmo básico para representar un árbol de decisión que utiliza la técnica de partición, se puede ver en la ilustración 3.Como lo muestra el algoritmo, usualmente se selecciona un nodo inicial que permite la clasificación de los ejemplos. Entonces, ¿Cómo se escoge el nodo raíz del árbol y los nodos en cada una de sus hojas? Según (12), un buen indicador para ello es la *Ganancia de información.* Para describir la ganancia de información es preciso adentrarse en el concepto de la entropía, se puede decir que la entropía está definida como:

$$Entropía(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus},$$
$$p_{\oplus} es\ la\ proporción\ de\ ejemplos\ positivos,$$
$$p_{\ominus}\ es\ la\ proporción\ de\ ejemplos\ negativos$$

Donde S es la colección de ejemplos positivos y negativos de algún concepto. Cuando le entropía es igual a 1, significa que se obtuvo igual número de ejemplos positivos y negativos. En caso que el número de ejemplos sea diferente el valor se encontrará entre 0 y 1.

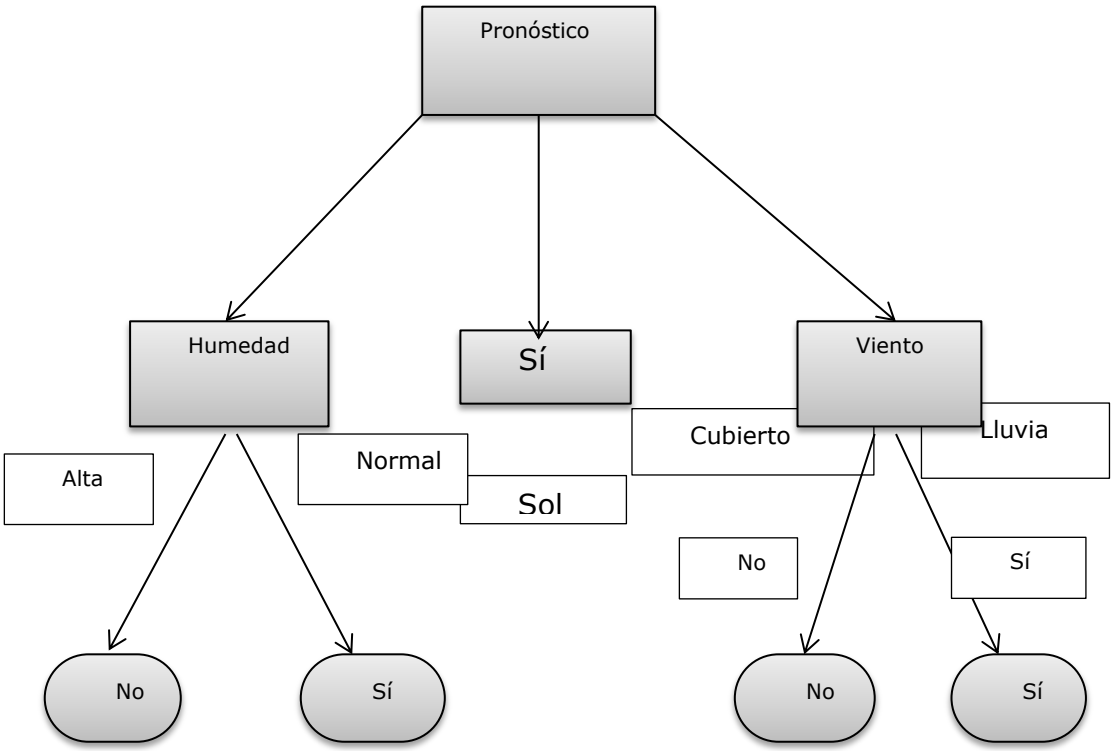


Ilustración 2 Ejemplo de un árbol de clasificación (12)

Por ejemplo, si p+ es igual a 1 la entropía es 0. Si p+ es 0.5, se tiene igual proporción de ejemplos positivos y negativos por lo que la entropía será máxima, ya que no se puede diferenciar si el mensaje pertenece a un ejemplo positivo o negativo. Con esta medida se puede entender el concepto de la ganancia de la información. *La Ganancia de Información* mide la efectividad de un atributo al clasificar datos. Esta se explica a través de la fórmula (12):

$$Ganancia(S,A) \equiv Entropía(S) - \sum_{v\,\epsilon\,Valores(A)} \frac{|S_v|}{|S|} Entropia(S_v)$$

Donde valores (A), es el conjunto de posibles valores para un atributo A, y Sv, es un subconjunto de S donde un atributo de A tiene un valor v. La Ecuación está dividida en dos términos: el primero calcula la entropía de la colección S. El segundo es un valor esperado de la entropía de S después de haber sido particionado usando el atributo A. La clave aquí es el valor esperado, pues permite medir la contribución de cada atributo en la partición. Por lo tanto la ganancia de información, es la reducción esperada en la entropía causada por conocer el valor de un atributo, de esta manera se hará la selección de nodos del árbol, aquel atributo que obtenga una mayor ganancia se utilizara como nodo raíz en cada paso de construcción del árbol.

Entrada: *Conjunto de Datos de entrenamiento, E*

***Salida**: Árbol de clasificación (T)*

Inicio

Si *Todos los Casos en E son de la misma clase* C_j

Entonces

Nodo simple etiquetado como C_j

Sino

Inicio

Seleccionar un atributo X_i*, con valores* $x_{i1}, \ldots . x_{in}$

Particionar E en $E_1, \ldots, E_n$ *de acuerdo con los valores de* X_i

Construir subárboles $T_1, \ldots, T_n$ *para* $E_1, \ldots, E_n$

Se tiene un árbol con raíz X_i *y subárboles* $T_1, \ldots, T_n$

Las ramas entre X_i*, y los subárboles se etiquetan como* $x_{i1}, \ldots . x_{in}$

Fin

Fin

Ilustración 3 Algoritmo básico de un árbol de decisión (11)

Los modelos de árboles de decisión tienen como objetivo obtener un modelo consistente y completo de acuerdo a la evidencia proporcionada. Los problemas que se pueden encontrar son:

1. Un modelo demasiado ajustado a la evidencia produce un error al introducir nuevos ejemplos. Lo anterior, no permite que el modelo sea lo suficientemente general.

2. En caso de que la evidencia contenga ruido, el modelo se verá mucho más afectado por ser demasiado específico, lo cual hará que el modelo se ajuste a los errores y perjudicará el modelo final.

3. La solución a estos problemas se puede encontrar a través de un ajuste al algoritmo para obtener un modelo más general y no tan específico. Este ajuste consiste en recortar el árbol, es decir, literalmente "podarlo", por lo cual se elige un límite de profundidad del árbol y se eliminan las ramas que estén por debajo del límite escogido.

4. Para la realización de la poda, existen dos técnicas que son resumidas brevemente a continuación: el pre poda, que consiste en realizar la poda antes de generar el árbol. la post-poda, como su nombre indica, consiste en realizar la poda después de construir el árbol. Esta técnica es muy común para mejorar el rendimiento de los árboles de decisión. También, existen otras técnicas como la restructuración (13), la transposición, la

transposición recursiva y la poda virtual. Estas técnicas tienen como objetivo brindar una mejor predicción al árbol de decisión.

Dependiendo de los criterios de partición y otras extensiones, se han desarrollado varios algoritmos. A continuación se mencionan los más importantes:

Algoritmo	Estrategia	Partición	Poda
CART (13) y derivados	divide y vencerás	GINI	Estimación dela complejidad de error
ID3 (2), C4.5 y otros	divide y vencerás	GAIN RATIO	Basada en Reglas
AQ (14), CN2	Métodos de cobertura.	BEAM SEARCH	Basada en umbrales
SLIQ (15)y Sprint (15)	Árbol de decisión clásico ajustado a grandes volúmenes de datos	GAIN	Basada en umbrales

El árbol de decisión es una técnica muy usada en la minería de datos, por sus múltiples aplicaciones y ventajas, que se describen en (11):

- Amplia aplicación a tareas de minería de Datos, como agrupamiento, clasificación, estimación de probabilidad
- Capacidad de tratar con tipos de datos numéricos y nominales
- Facilidad de uso.
- Eficiencia y capacidad de manejar grandes volúmenes de datos

2.5 El problema de los datos estructurados

Anteriormente, se ha hecho referencia de los diferentes métodos para aprender sobre los datos escalares (nominales y numéricos), pero esta no es la única forma de representar la información del mundo real. En diversos escenarios, como la biología, la química, la informática, etc… encontramos información que ha sido sintetizada como otro tipo dato: los datos estructurados. Estos permiten a manera de listas, secuencias o grafos representar información. En consecuencia los datos estructurados, son una de las áreas de investigación en aprendizaje automático. Según (16 pág. 23), la importancia de la investigación de los datos estructurados está en su representación, esta no es lo suficientemente expresiva cuando los datos tienen una estructura interna, ya que están compuestos de sub partes altamente relacionadas. Como ejemplo para entender el problema de los datos estructurados, se propone un problema clásico: el desafío Este-Oeste, propuesto por R.S. Michalsky. Se tiene un conjunto de datos de entrenamiento que contiene 10 trenes, donde cada uno de los trenes tiene un número variable de vagones en un orden cualquiera. Cada vagón, es representado por un número fijo de características (tamaño, peso, forma, número de ruedas, tipo de techo, tipo de carga) en cada turno. Además, cada vagón es etiquetado de acuerdo con la dirección que toma (este-oeste). El objetivo es encontrar el clasificador que indique la dirección del tren, una solución posible puede ser "un tren se dirige al oeste si tiene 10 ruedas y su forma es ovalada".

En la práctica muchos de los problemas planteados en la vida real tienen esta configuración y muestran un escenario de aprendizaje particular llamado "aprendizaje de múltiple instancia". De esta forma, los ejemplos son representados como un conjunto de tuplas, no como un único elemento, donde una tupla será la responsable de la clasificación. Se hace necesario para trabajar con los tipos de datos estructurados, saber cómo pueden ser representados. A diferencia de las representaciones proposicionales donde los datos consisten de longitud fija de tuplas con valores numéricos o nominales, los datos estructurados se pueden representar de distintas formas como: conjuntos, listas, grafos. Para realizar una aproximación a la representación de datos estructurados es importante hace referencia a las distintas maneras en que los datos estructurados son representados. Una de esas aproximaciones según (7 págs. 23,24), es la representación basada en la lógica de primer orden, esta permite representar objetos complejos y si es necesario establecer reglas y relaciones que los objetos deben satisfacer. Una de las características de la lógica de primer orden es su alta expresividad, por lo cual solo un subconjunto de esta se utiliza para labores de aprendizaje.

Algunas de las aproximaciones usadas para la representación son:

1. Representación ISP(*Individual-Structural-property*):Esta representación es un subconjunto basado en Prolog. Los átomos se utilizan para describir objetos y éstos se agrupan en tres categorías: individuales, lo que indica el tipo de objeto, estructural, que representa las relaciones entre sub-partes de un objeto y la propiedad, que describe estas subpartes .En todo tipo de datos, se distinguen dos secciones. La primera,

donde los predicados para ser utilizados por las descripciones de los objetos se declaran (parte declarativa). La segunda, contiene las descripciones de los objetos. Una de las ventajas derivadas de esta es la simplificación de los operadores de generalización/especificación utilizados por los algoritmos de aprendizaje. También, el ISP permite la consideración de diferentes puntos de vistas individuales sin cambiar la representación. La mayor desventaja, se encuentra en la perdida de estructura, ya que las relaciones entre partes/subpartes debe ser representada explícitamente

2. Representación basada en términos: Esta técnica permite hacer una representación más compacta de los objetos. Aquí los objetos y subpartes están representados por formulas recursivas tan complejas como el problema que se menciona. Es decir el símbolo de predicado sólo es la que se refiere el objeto mientras los funtores se refieren a la (sub) partes de los objetos y las constantes de referirse a las propiedades de estas partes (sub). La consecuencia más significativa de este tipo de representación es la necesidad de usar predicados adicionales para extraer la información empacada en estos términos.
3. Representación basada en bases de datos relacionales y lógica de alto Orden: Para el aprendizaje automático la representación de complejas estructuras de datos basada en los casos anteriores ha sido muy exitoso. Aun así, las bases de datos relacionales y la lógica de alto orden son ampliamente usados para la representación de estructuras de datos complejos. Su utilidad está dada por la capacidad de manejar funciones y átomos de manera uniforme a través del uso de funciones booleanas, que permiten funciones anidadas. Por ejemplo una expresión como "los trenes tiene al menos un vagón corto" puede ser presentada a través de la fórmula:

$$Existe(T, longitud(.) = corto),$$

Donde *Existe,* es una función de alto orden que devuelve verdadero si existe un elemento C de T (donde T y C representan al tren y al carro respectivamente) que satisfacen la condición longitud (C)=corto.

Una vez que se han mencionado algunas posibles formas de presentar los datos estructurados es necesario conocer los métodos con los cuales se pueden reconocer patrones. Estos métodos serán los que se exponen en el siguiente numeral.

2.5.1 Aprendizaje Simbólico sobre Datos Estructurados

El marco de trabajo del aprendizaje simbólico en datos estructurados, ha sido cubierto por la lógica de primer orden, que como se vio anteriormente permite realizar hipótesis adecuadas sobre su representación. Este a su vez, está altamente relacionado con la programación lógica inductiva (ILP), la inferencia gramatical y la extensión de los árboles de decisión. Aunque no es el objetivo de este trabajo se nombran a continuación algunas de las metodologías más utilizadas en este dada su importancia para el problema de los datos estructurados:

1. ILP(Programación-Lógica-Inductiva):Es más conocida a través de su lenguaje de programación Prolog. Esta permite realizar algoritmos bien fundados, la formalización hace que sea cercano al lenguaje natural por lo cual la hipótesis puede ser directamente interpretada por los usuarios. Las ventajas presentadas por la ILP son variadas: su alta expresividad, resultados técnicos y teóricos tomados de la programación lógica. Los métodos basados en ILP se puede clasificar en dos categorías: método bottom-up, y métodos top-down.

2. Proposicionalizacion: La proposicionalizacion es un conjunto de técnicas que permiten la transformación de problemas relaciones en problemas representados por atributos valuados.

3. Aprendizaje de datos multi-relacionales y lógica de alto orden: Debido a que las bases de datos relacionales y la representación ISP están altamente relacionada, el uso de los métodos de aprendizajes basados en ILP se vuelve importante para el descubrimiento de patrones. Básicamente el uso de test booleanos usados en sentencias SQL es un patrón multi-relacional, como es comentado en (7 pág. 40). También se puede ver que la Lógica de alto orden es otro formalismo más para la representación de datos estructurados donde un árbol de decisión pude ser presentado como un test booleano que representa fórmulas de alto-orden, como lo menciona (7 pág. 41).

4. Aprendizaje gramatical desde secuencias: Las secuencias son estructuras muy estudiadas en aprendizaje automático, pues son encontradas en múltiples escenarios como teoría del control, extracción de patrones, traducción de idiomas o biología computacional. El problema estudiado en las secuencias es conocido como la inferencia gramatical, donde aprender la gramática es el objetivo principal partiendo de conjuntos de ejemplos positivos y negativos. Una de las técnicas más conocida, es la RNPI(Regular positive-Negative inference) y su objetivo es lograr a partir de una generalización mezclar estados hasta cubrir un ejemplo negativo.

2.5.2 Aprendizaje No Simbólico sobre Datos Estructurados

Para el aprendizaje no simbólico este apartado se ocupa de las diferentes aproximaciones, enfocándose en el aprendizaje basado en similaridad, específicamente cuando las funciones están basadas en distancia.

Los métodos basados en distancia, como se menciona anteriormente usan la noción de disimilaridad para ser adoptada en un dominio que contiene datos estructurados. Esa noción de distancia es aplicada con el fin de aplicar una métrica fiable entre datos no estructurados. Por esto a continuación se hace un resumen de las diferentes distancias usadas en cada uno de las estructuras de datos más comunes. Estas son:

- Conjuntos: En los conjuntos la métrica más usada se presenta por $|(A-B) \cup (B-A)|$, donde A y B son conjuntos finitos Esta función de distancia cumple con las propiedades de identidad, simetría y desigualdad triangular. La distancia entre los dos conjuntos está dada por el número de elementos que tienen en común. Si se toma como ejemplo los siguientes conjuntos de secuencias $A = \{bc, b^4\}\, y\, B = \{bc, d^4\}$ y $C = \{bc, d^3\}$ entonces $d(A,B) = d(A,C) = 2$, la diferencia simétrica consiste que cualquier elemento en el conjunto es igualmente diferente al resto de ellos. Otro método para medir la distancia, de manera más precisa es cuando se aplica la distancia de Hausdorff que es define como:

$$d_H = \max\{sup_{a \in A}\, inf_{b \in B}\, d(a,b)\,, sup_{b \in B}\, inf_{a \in A}\, d(a,b)\}$$

  Donde A y B son dos conjuntos y d es una distancia definida sobre los elementos. Algunos autores proponen una familia de semi-distancias cuando una función de similaridad sobre los elementos ha sido definida previamente. Una formalización de esta idea se puede encontrar en (7 pág. 43).

- Listas: La distancia de Hamming, es una métrica capaz de establecer un código para la detección y auto-corrección de códigos. Esta usa una lista de igual longitud donde la distancia calculada es el número de posiciones para la cual el símbolo correspondiente es diferente. Por ejemplo, dada una secuencia s1 = aabb y s2= accb, entonces d(s1,s2) = 2. La distancia que más ha sido usada dentro de la literatura, es la distancia de edición (Levenshtein), que tiene por objetivo contar el menor número de operaciones (inserción, borrado, sustitución) para transformar una cadena en otra. Otra distancia para calcular la distancia entre listas es la propuesta por (17).

- Átomos: Una distancia para átomos está definida por una medición de distancia limitada (0<d<1), donde se toma la profundidad de ocurrencia de los símbolos de tal manera que el resultado de la diferencia más cercana al núcleo de símbolos tiene un mayor valor. Por ejemplo dados 2 átomos $e_1 = p(s_1, \ldots., s_n)$ y $e_2 = q(t_1, \ldots., t_n)$, esta distancia se define recursivamente así:

$$d(e_1, e_2) = \begin{cases} 0, & si\ e_1 = e_2. \\ 1, & si\ p \neq q. \\ \dfrac{1}{2_n} \displaystyle\sum_{i=1}^{n} d(s_i, t_i),\ por\ lo\ demas \end{cases}$$

  Esta función devuelve un par ordenado de valores enteros (i,j). Esta pareja de valores expresa la diferencia entre dos átomos2, basados en componentes: i, (la función de símbolos) y j,(variable de símbolos). Esta distancia es citada con más detalle en (7 pág. 45.)

- Grafos: Según (7 pág. 47), una función de distancia para grafos etiquetados es implementada. Para poder exponerla es necesario un explicación previa, un grafo es un función $g(V, \alpha, \beta)$ donde es V es un conjunto finito de vértices $\alpha: V \rightarrow L$ y $\beta: V * V \leftarrow L$ son nodos y una función para etiquetar los bordes, respectivamente. Asi los grafos son siempre completos ya que los bordes perdidos recién una etiqueta de valor Null. Además, existe el concepto de máquina de corrección de errores en grafos (*ecgm*, por sus siglas en ingles), que permite hacer una serie de operaciones de edición donde se puede transformar un grafo en otro, y el costo asociado a un *ecgm.* En esta medición la distancia entre dos grafos g1 y g2 es el mínimo costo tomado por de todos los ecgm calculados. Un ejemplo completo se puede ilustrar en (7 pág. 48).

En la mayoría de los métodos basado en distancia, el aprendizaje es independiente de su representación de los datos. Por lo tanto, el aporte al trabajar con estas estructuras de datos complejas se da en el momento en que se pueden extraer los patrones simbólicos de cualquier representación de datos a partir de una función de similaridad. Por la tanto, la idea de crear un marco de trabajo que permita a partir de una función de similaridad dada por una distancia y unos datos representados por un espacio métrico, son los expuestos en el trabajo de (7).Este marco de trabajo es el utilizado en esta disertación con el fin de profundizar en el aporte ya realizado.

3 Algoritmos.

Loa algoritmos de minería de datos proporcionan un conjunto de cálculos y reglas, con el fin de analizar los datos y extraer patrones específicos o tendencias. De acuerdo con las diferentes técnicas expuestas en este trabajo, se pretende realizar una comparación de diferentes algoritmos para trabajar conjuntos de datos con tipos de datos numéricos, nominales y estructurados. Para hacer esta comparación se escogieron los algoritmos C4.5, Knn, árboles de decisión y un nuevo método conocido como DBDT. En las siguientes páginas se explican los diferentes algoritmos.

3.1 KNN

El KNN es una de las técnicas que permite realizar clasificación a partir de conjuntos datos sin necesidad de construir modelos previamente. El objetivo de las técnicas de aprendizaje retardado (lazy) es realizar muy poco esfuerzo al momento de aprender, por lo cual no generan modelos con los datos de entrada, sino que almacenan los ejemplos.

El algoritmo KNN(K nearest neighbors) es uno de los algoritmos basados en distancia más conocido y utilizado. Su función consiste en clasificar una instancia de la clase más fuertemente representada en sus K vecinos más próximos.

Según (18) el algoritmo de los vecinos más próximos se puede expresar como:

- *Algoritmo de entrenamiento: Para cada ejemplo de un conjunto de datos* $(x, f(x))$*, se agrega el ejemplo al conjunto de entrenamiento.*
- *Algoritmo de clasificación:*
  - *Dada un instancia* x_q *a ser clasificada en un conjunto de clases.*
  - *Sean* $x_1 \ldots\ldots x_{k,}$ *que denotan las k instancias de un conjunto de entrenamiento, más cercanas a* x_q
  - *Entonces*,

$$f(x_q) \leftarrow argmax_{\,v \in V} \sum_{i=1}^{k} \delta(v, f(x_i)$$

$$donde\ \delta(a,b) = 1\ si\ a = b, sino\ \delta(a,b) = 0$$

Este algoritmo, logra que la clase más votada de los k vecinos más próximos sea la clase asignada. En el caso de aparecer un empate, se debe definir una regla de resolución, como por ejemplo: clasificar la clase por medio de su vecino más próximo entre las que se encuentra el empate, seleccionar la clase haciendo una ponderación de las distancias de sus vecinos… etc.

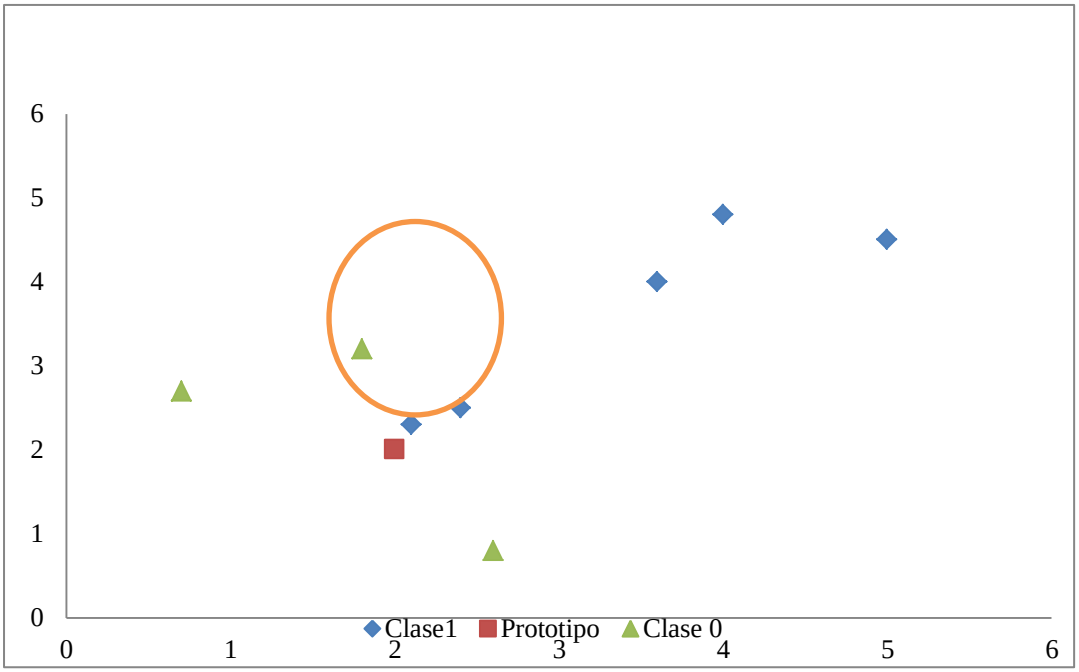


Ilustración 4, un ejemplo de KNN con 2 clases (k=3)

El método de los vecinos más próximos (K-NN) a diferencia del método NN (19) tiene varias ventajas: permite aprovechar de manera más eficiente la información del conjunto de entrenamiento. También, Mitchell (18) señala, la robustez y la efectividad en grandes conjuntos de datos. Una de las mejoras más significativas en este algoritmo es la implementación de una distancia ponderada en la función, así:

$$f(x_q) \leftarrow argmax_{\,v \in V} \sum_{i=1}^{k} w_i \delta(v, f(x_i)$$

$$donde\ w_i \equiv \frac{1}{d(x_q, x_i)^2}$$

Al realizar esta modificación se da una mayor contribución a los vecinos más próximos de la instancia que se está clasificando. Además, permite que los ejemplos que estén más distantes de la instancia a clasificar tengan un menor efecto, como podemos ver en la ilustración 4.

Los vecinos más próximos, se utilizan normalmente para la clasificación de tipos de datos numéricos o nominales. En este trabajo, se ha modificado para poder manejar también atributos de tipos estructurado. Esa modificación se realizó utilizando la distancia de edición para medir la distancia entre cadenas. Esta extensión del KNN, se ha llamado KNN-S. El KNN-S permite al momento de la clasificación, si el atributo es del tipo estructurado, aplicar la distancia de edición normalizada, para contribuir de la misma manera como los demás elementos de los otros atributos. Esta extensión nos permite realizar comparaciones con diferentes algoritmos que tengan la misma característica al manejar atributos tipo estructurado.

3.1.1 Adaptación del KNN para el manejo de cadenas (KNN-S)

EL KNN es uno de los algoritmos más utilizados en la minería de datos su sencillez y la capacidad de trabajar sin un modelo entrenado, lo hacen una técnica de clasificación muy utilizada. Uno de las grandes desventajas, es el hecho de no manejar con datos estructurados, sino atributos únicamente nominales en los cuales clasifica las instancias diferentes como uno o cero si son iguales. En este trabajo se busca superar esta dificultad con el fin de tener un algoritmo que permita comparar, en condiciones similares con otros algoritmos que usan datos estructurados.

Con el fin de ayudar a la clasificación de los datos estructurados se realizó un ajuste al KNN que consiste en introducir la distancia de Levenshtein, en el caso que se necesite clasificar una nueva instancia que contenga datos estructurados como cadenas. Así el algoritmo utilizara la distancia de edición para ponderar el valor del dato estructurado. Esta modificación se realizó en el algoritmo de clasificación KNN, y se puede ilustrar el funcionamiento así:

- *Algoritmo de clasificación:*
    - *Dada un instancia x_q a ser clasificada en un conjunto de clases.*
    - *Sean $x_1 \ldots\ldots x_{k,}$ que denotan las k instancias de un conjunto de entrenamiento, más cercanas a x_q*

- *Entonces*,

$$f(x_q) \leftarrow argmax_{\,v \in V} \sum_{i=1}^{k} \delta(v, f(x_i)$$

$$donde\ \delta(a,b) = 1\ si\ a = b, sino\ \delta(a,b) = 0,$$

$$donde\ \ DL(a,b), si\ a\ \ y\ b\ \in string.$$

Esta modificación se realizó utilizando la implementación de WEKA, y haciendo una extensión de la clase que implementa el KNN, la descripción del código se puede encontrar en el apéndice C. El KNN evalúa las instancias una por una realizando una sumatoria de los valores calculados, según el tipo de cada atributo. Por ejemplo, si tenemos una cadena A={32,H20,aceptable} y una cadena B={21,C02,deficiente}, donde se tiene un atributo de tipo numérico, cadena y nominal respectivamente se espera que la distancia entre atributos sea C={11,2,1}, donde el resultado es el cálculo de la distancia respecto al tipo de dato correspondiente. En la ejecución del KNN se van procesando las distancias para cada atributo en esta caso las distancias de los atributos estructurados son pre-procesadas y se incorporan a un matriz de distancia como el ejemplo XXX. Una vez las distancias de las cadenas están en la matriz, se toma su valor y añade al cálculo total de las distancias en algoritmo de clasificación. Se realizó así con el fin de disminuir el tiempo de procesamiento del algoritmo de clasificación pues resulta más eficiente realizar la búsqueda en una matriz que calcular las distancias en tiempo de ejecución.

Se decidió nombrar esta mejora del KNN como KNN-S, con este ajuste se espera obtener un clasificador que permita evaluar en las mismas condiciones los tres tipos de datos junto a los demás algoritmos aquí expuestos.

3.2 C4.5

El C4.5, es uno de los algoritmos más populares en la minería de datos. Esto se debe a su fiabilidad y capacidad para representar reglas del modo *si-entonces*, que permiten una mayor comprensibilidad de los resultados. El C4.5, es una extensión mejorada del algoritmo ID3, que se ha mencionado en el capítulo anterior como uno de los algoritmos clásicos para construir arboles de decisión. En (12), se describen las mejoras de este algoritmo:

- Capacidad de tratar atributos numéricos y no solo nominales, como el ID3

- *Evitar el sobreajuste de los datos que se da producto del incremento del ruido o existe con muy pocos atributos*. El sobreajuste se puede evitar según (18) a través de 2 aproximaciones. La primera es reduciendo el tamaño de los arboles antes de llegar al punto donde esté completamente clasificado, lo cual

es una tarea difícil de estimar. La otra aproximación es permitir que exista el sobreajuste y después podar el árbol de decisión.

• *Reducción de la poda de error*: esta aproximación requiere evaluar si cada nodo del árbol de decisión es candidato a ser podado. El nodo será podado si es el que menos contribuye a la precisión en el árbol.

• *Regla Post- poda*, como su nombre lo indica permite podar el árbol de decisión una vez construido. Una vez se ha construido el árbol y se ha convertido a su conjunto de reglas correspondiente, se generaliza cada regla removiendo las precondiciones que aumentan la precisión. Una vez realizada la generalización se organizan las reglas de acuerdo a su precisión.

•*Medidas alternativas para seleccionar atributos*: esta mejora del algoritmo original tiene por objetivo reducir el error cuando los atributos tienen mucha información en un conjunto de datos. Al tener mucha información la medida de ganancia de información favorece a los atributos que tienen mayor información, aunque estos no estén relacionados directamente con el objetivo de la tarea a realizar. Lo anterior, se puede solucionar a través de la elección de otra medida como el Ratio de Ganancia (Gain Ratio), que permite a través de un término conocido como división de la información (Split Information), medir la entropía de un atributo respecto a otro atributo como lo muestra la siguiente ecuación:

$$Split\ Information(S,A) = -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

Donde S1 hasta Sc, son los subconjuntos de ejemplos resultantes de la partición S del valor c evaluado en el atributo A. La división de la información equilibra la contribución de los atributos no por la extensión de sus valores sino por su aporte, por esto la ganancia de información lo vinculación a su ecuación como se muestra a continuación:

$$Gain\ Ratio(S,A) \equiv \frac{Gain(S,A)}{SplitInformation(S,A)}$$

Como indica la ecuación, la división de la información es inversamente proporcional a la ganancia de información, con lo cual se mantiene el equilibrio al momento de evaluar cada atributo, respecto a su contribución y no la cantidad de información por atributo.

- *Manejo de ejemplos de entrenamiento con valores perdidos en los atributos*. De manera frecuente, los conjuntos de datos suelen tener valores no establecidos o perdidos en los conjuntos de datos. Por ejemplo, las bases de datos de carácter comercial suelen no tener los datos completos de sus clientes (edad, sexo, etc…), por lo cual es necesario estimar el valor de esos datos, para poder evaluar la ganancia de la información, para ello existen 2 estrategias.

La primera, es dar el valor que más veces aparezca en ese atributo, por ejemplo en el caso de ser el atributo edad y la edad que más se repite es 28 entonces se asignará 28 a los valores faltantes del atributo edad; la segunda, es estimar la probabilidad de cada uno de los valores del atributo.

- *Manejo de atributos con costos diferentes*: los conjuntos de datos suelen tener atributos con diferentes costos asociados. En el ejemplo de la base de datos de carácter comercial pueden aparecer atributos tales como valor, cantidad, moneda…etc.

3.3 DBDT

Los métodos basados en distancia se pueden clasificar según su tipo de datos (ID3 solo maneja atributos nominales, c4.5 maneja nominales más numéricos). También, existen métodos más complejos que se encargan de datos relacionales como el TILDE (20) o FOIL (21), hasta el momento ninguno trata con tipos de datos como cadenas, árboles y grafos directamente. Aunque estos métodos que manejan datos estructurados no lo hacen de una manera óptima, se han realizado mejoras. Por ejemplo, una extensión del c4.5, donde se requiere definir las particiones y se exige manejar particiones para cada tipo de datos. Esta posible solución, obliga a restringir las particiones para cada tipo de dato o seleccionar previamente qué tipo de particiones resultarían útiles para cada problema en particular, omitiendo así la generalización del problema en particular. Al ser el DBDT una propuesta basada en distancia, usa un método muy similar al de partición por centros (centre splitting).Este método, consiste en dividir el espacio métrico en diferentes regiones, donde cada región es representada por un punto que es llamado centro. Cada vez que se realiza el proceso, se calcula el centro de acuerdo a cada clase del conjunto de datos. El valor calculado para cada ejemplo de cada clase puede ser igual al centro de su respectiva clase o no. Cada ejemplo calculado se asocia con el punto más cercano. El proceso finaliza cuando las regiones se vuelven puras, lo cual indica que todos los ejemplos pertenecen a una clase. Esta aproximación, es similar a las aproximaciones clásicas de los árboles de decisión. Es novedosa, en cuanto el manejo de tipos de datos estructurados. Para poder manipular este tipo de datos, hay que determinar cómo son computados los centros para esto tipos de datos.

La regla para el cálculo de los centros será: "Si se quiere un centro por clase, la primera idea será calcular el centro el cual minimiza la distancia para cada uno de los demás ejemplos de cada clase", según lo afirma (7). Por ejemplo para una instancia dada $\{[c],[c,c],[c,c,c,c]\}$, el centro está dado por una lista que contiene 4c.

Esta aunque no aparezca explícitamente en la lista es el centro, por lo cual se buscara alguno que sea razonablemente más representativo que en este caso seria $[c,c,c,c]$. Para introducir el algoritmo del DBDT es importante definir una notación, como aparece a continuación:

- $Attr_{xj}(e)$ *devuelve el* $j-esimo$ *atributo del ejemplo* e.
- $Clase(e)$ *devuelve la clase del ejemplo* e.
- $Valores\ (S, xj)$*devuelve el # de valores* $\neq$ *para el atributo* xj *en el ejemplo* S.
- $Card(v, S, xj)$ *devuelve el # de ocurrencias de un valor* v *en un atributo*
- xj *en el ejemplo* S
- $Dist(x, y)$ *calcula distancias entre valores* x *e* y *para atributos que son del mismo tipo.*
- *Se asume que las distancias estan precalculadas.*

Entrada: S es un conjunto de ejemplos de la forma $(x1, \dots, xn), n > 1$,m es el máximo número de hijos por nodo.
Inicio
$C \leftarrow \{Clase(e):\ e \epsilon S\}$
$Si\ |C| < 2\ entonces\ Salir$
Para cada $atributo\ xj$ **haga**
SI $Valores\ (S, xj) < 2$ **entonces** CONTINUE;
$protList \leftarrow CalculaPrototipos(xj, S, m, C)$;
Si $Tamaño(ProtList) \leq 1$ **Entonces** Salir;
$Split\ \ j \leftarrow 0$;
Para $i = 1 \dots longitud(ProtList\ )$ **haga**
$\widehat{Si} \leftarrow \{e \in S:\ i = Atrae(e, ProtList, xj)\}$ // Ejemplos atraídos por i;
$Split\ \ j \leftarrow Split\ j \cup \widehat{S_i}$; // Se agrega un nuevo nodo;
Fin
Fin
$BestSplit \leftarrow ArgMaxSplit_j(Optimality(Split_j))$
Para cada $set\ S_k \in BestSplit$ **haga** $DBDT\ (S_k, n)$ □
Fin

Ilustración 3 Algoritmo DBDT

Como muestra la ilustración 3, es un árbol decisión común que determina para cada atributo, una lista de prototipos valorados, el cual compondrá un conjunto de hijos para cada nodo. La diferencia con los arboles de decisión típicos se hacen con dos funciones:

- Función Atrae: esta función determina cuál prototipo es asignado a un nuevo ejemplo. Esta función también se utiliza cuando el árbol de decisión está aprendiendo o siendo usado para predecir nuevos valores. La finalidad de la función es obtener el prototipo más cercano, en caso de estar empatados devuelve el prototipo que se encuentre más a la derecha.

Entrada: S es un conjunto de ejemplos de la forma $(x1, \dots, xn), n > 1$,m es el máximo número de hijos por nodo.

Inicio

$C \leftarrow \{Clase(e):\ e\ \epsilon\ S\}$

$Si\ |C| < 2\ entonces\ Salir$

Para cada $atributo\ xj$ **haga**

SI $Valores\ (S, xj) < 2$ **entonces** CONTINUE;

$protList \leftarrow CalculaPrototipos(xj, S, m, C);$

Si $Tamaño(ProtList) \leq 1$ **Entonces** Salir;

$Split\ \ j \leftarrow 0;$

Para $i = 1 \dots longitud(ProtList\ )$ **haga**

$\widehat{Si}\ \leftarrow \{e\ \in\ S:\ i =\ Atrae(e, ProtList, xj)\}$ // Ejemplos atraídos por i;

$Split\ \ j \leftarrow Split\ j\ \cup\ \widehat{S_i}$; // Se agrega un nuevo nodo;

Fin

Fin

$BestSplit\ \leftarrow ArgMaxSplit_j(Optimality(Split_j))$

Para cada $set\ S_k\ \in\ BestSplit$ **haga** $DBDT\ (S_k, n)$ ⬚

Fin

Ilustración 4 DBDT función Atrae

- Función Calcula prototipos: esta función se puede realizar de muchas maneras diferentes. Vamos a suponer que el objetivo es obtener el mejor prototipo. Para esto, removemos la clase y el valor del atributo y buscamos el mejor prototipo siguiente, una y otra vez hasta llegar al límite M o termine el número de clases o los valores de los atributos. Esta función es la más importante, si la dividimos en dos partes, la primera (donde aparece la primera instrucción for), donde $\vartheta(|C| \cdot |V_c| \cdot |V_c|)$ donde V_c es el número de valores diferentes para el atributo. Así, este orden es lineal $|C|$ y cuadrático en V_c, la cardinalidad de V_c puede o no ser mínima para algunos atributos estructurados y nominales. En el caso de los atributos estructurados y para el DBDT, las distancias se pre calculan antes de llamar al procedimiento DBDT, mostrado anteriormente. En la segunda parte (la siguiente instrucción *for*) tiene un *argmin* que puede ser calculada como $\vartheta(\#Prots \cdot |V_c| \cdot |V_c|)$ que generalmente es más pequeño que la primera parte. Cabe anotar que las distancias son calculadas entre atributos y no entre ejemplos, lo cual hace que su eficiencia sea mucho mayor.

El DBDT, es una nueva propuesta, una aproximación basada en distancia para arboles decisión; donde se pretende realizar particiones de cualquier tipo de dato, usando los prototipos de cada clase. El resultado del mismo, será un árbol de decisión donde se muestran los diferentes nodos y su similaridad representada por la distancia entre los mismos. Con el fin de obtener mayor comprensión del mismo, se explicara a continuación el método utilizado por el DBDT

Entrada: e es un ejemplo, Protlist es una lista de prototipos clasificados y Xj un atributo escogido.

Salida: Índice del prototipo que atrae e;

Inicio

Para i=1, …, length(ProtList) **haga**

$v \leftarrow attr_{x_j}(e)$

Si $\forall k > i, distance\left(attr_{x_j}(ProtList[i]), v\right) distance\left(attr_{x_j}(ProtList[k]), v\right)$

Entonces

Devolver i;

FIN

FIN

FIN

Ilustración 5 DBDT función Calcula Prototipos

3.3.1 Adaptación del DBDT para procesamiento de cadenas.

Como se ha mencionado anteriormente el DBDT, puede procesar información relacionada a diferentes estructuras de datos. Para este ejercicio, se realizó el procesamiento de secuencias o cadenas: las secuencias tiene como característica principal una codificación en la cual se puede aplicar la distancia de Levenshtein con el objetivo de conseguir la distancia de una cadena a otra.

Para dar claridad a continuación se expone un ejemplo de cómo funciona la distancia de Levenshtein para el tratamiento de secuencias. La distancia de Levenshtein, es usada como una generalización de la distancia de Hamming y consiste en calcular el número de pasos que se necesita para convertir una secuencia A en una secuencia B.

Si se define A = "casa" y B= "calle", la transformación óptima (la más eficiente) para que A pueda convertirse en B consiste en:

1. casa → cala (sustitución de 's' por 'l')
2. cala → calla (inserción de 'l' entre 'l' y 'a')
3. calla → calle (sustitución de 'a' por 'e')

El ejemplo ilustra cómo funciona la distancia de Levenshtein, usando operaciones básicas como la inserción, sustitución y borrado. Podemos decir entonces que d (A, B) = 3, para la transformación óptima. Como se ha mencionado anteriormente, no hay una única transformación óptima puede existir más de una

transformación óptima. El ejemplo también se puede ilustrar usando una matriz de distancia como la siguiente:

	0	1	2	3	4
		C	A	S	A
1	C	0	1	2	3
2	A	1	0	1	2
3	L	2	1	1	2
4	L	3	2	2	2
5	E	4	3	3	3

Tabla 1.Matriz de distancia

La resultante de esta matriz, la posición (5,4)=3 que es el resultado que hemos visto anteriormente. La matriz busca la función más óptima para poder calcular la transformación de una secuencia en otra. Un matriz de distancia es una forma simple para representar un espacio métrico, el DBDT busca poder utilizar este espacio métrico para poder representar las distancia ya no entre palabras sino entre secuencias, sin importar su extensión que permiten reducir la complejidad de una secuencia a la distancia que existe entre ellas, este es la simple pero interesante estructura que maneja el DBDT.

Vamos a suponer que existen 5 secuencias para hacer la comparación de su similitud, para esto podremos utilizar un matriz como la anterior donde cada casilla muestre, ya no el valor de cada operación de transformación sino la distancia que existe entre cada secuencia el resultado se muestra en la tabla2.

	C3H6O	C8H8O	C8H6O4	C9H6O6	C8H6O2
C3H6O	0	2	2	2	2
C8H8O	2	0	2	3	2
C8H6O4	2	2	0	2	1
C9H6O6	2	3	2	0	2
C8H6O2	2	2	1	2	0

Tabla 2 Matriz de distancia para elementos químicos

Las matrices de distancia son una forma sencilla de abordar el problema de la distancia, pues una vez se tiene es más sencillo obtener la similitud entre las cadenas. Igualmente, el DBDT aborda la solución del manejo de atributos estructurados como cadenas reduciendo la complejidad al pre-procesar las distancias y ponerlas a funcionar en una matriz. Este mecanismo permite abordar el manejo de datos estructurados en la minería de datos de una manera ágil y novedosa.

El DBDT funciona como una extensión del software WEKA y su operación se puede dividir en dos etapas:

1. EL DBDT hace el pre procesamiento del conjunto de datos con el fin de obtener la matriz de distancia asociada para cada atributo
2. El DBDT ejecuta el algoritmo de usando el archivo de matrices de distancia y el conjunto de datos como para metros de entrada.

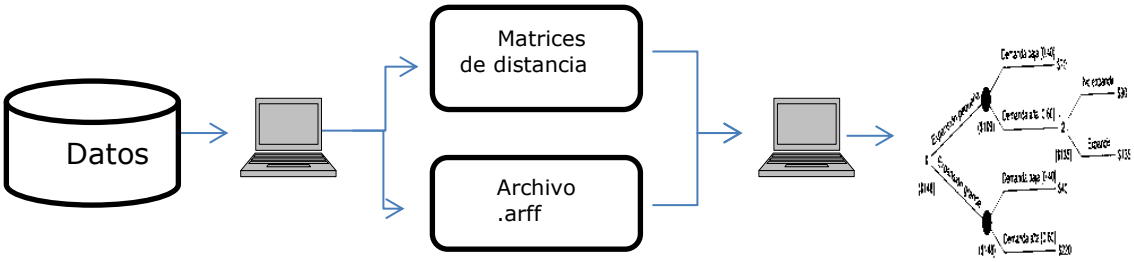


Ilustración 6 Arquitectura del sistema DBDT

3.3.2 Un ejemplo usando el DBDT

Con el fin, de ilustrar el uso del DBDT se describe a continuación un experimento de ejemplo tomado de (7 pág. 197). Anteriormente, se tomó como ejemplo de un conjunto de datos la Tabla 4. El objetivo para este conjunto de datos, es descubrir una seria de reglas o una función de clasificación que permita identificar, de manera comprensible, todos los grupos de moléculas.

Id	Temperatura(Cº)	Molécula	Clase
1	30	xc5	1
2	31	xc35b	1
3	29	xc35bc5bc5	1
4	33	yc27bc13bc10	1
5	34	yc33bc7 1	1
6	31	zc5	2
7	51	tc12	2
8	53	zc10	2
9	46	zc7	2
10	46	zc9	2
11	47	tc14	2
12	49	zc5	2
13	55	tc30bc10bc10	3
14	49	zc25	3
15	50	zc29b	3
16	48	tc30bc7	3
17	52	tc31b	3
18	53	tc28bc12	3

Tabla 3 Conjunto de datos con datos estructurados.

Como podemos ver en la Tabla2, las instancias están compuestas por un atributo numérico (Temp. (Cº)) y un atributo estructurado (molécula). El atributo 'temperatura' es un tipo de dato numérico y la molécula es sólo una lista finita de los símbolos pertenecientes a un alfabeto {b, c, x, y, z, t}; como se ha mostrado anteriormente es necesario generar un espacio métrico basado en distancia, así para la temperatura será la diferencia entre números reales y la distancia de edición (Levenshtein) será usada para la diferencia entre moléculas.

De acuerdo con el ejemplo ambos nodos temperatura y molécula son candidatos a ser el nodo raíz. Para este ejemplo, se tienen 3 centroides por atributo. La mejor partición es dada por el primer atributo, el algoritmo desglosa el árbol hasta tener nodos puros. Y se puede obtener un árbol de decisión como lo muestra la ilustración 7.

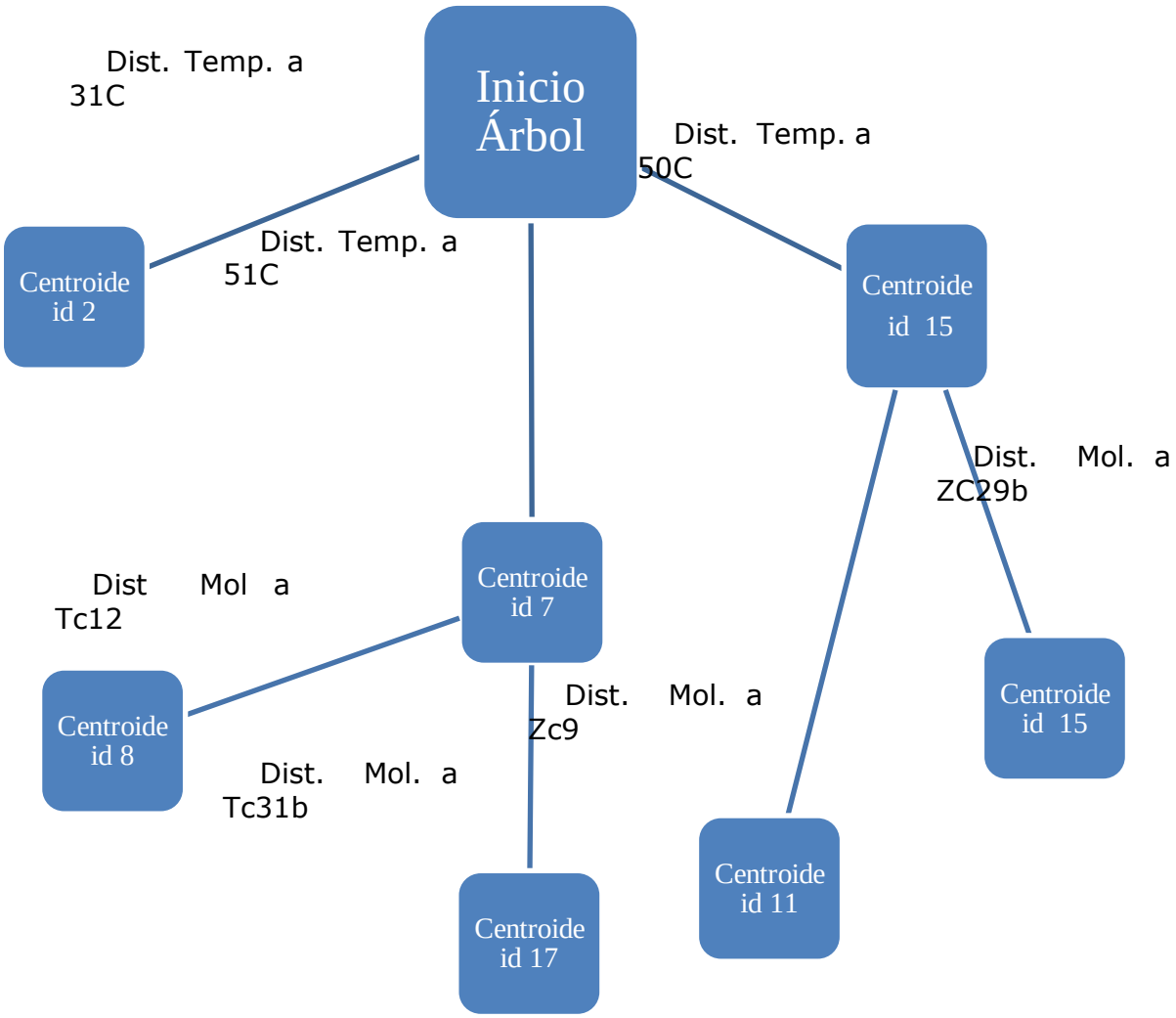


Ilustración 7 Ejemplo árbol de decisión

4 Análisis Comparativo

El análisis comparativo se realizó de acuerdo a los objetivos: la implementación de la distancia de edición, la escogencia de conjuntos de datos estructurados, realizando medidas de evaluación que permiten comparar el rendimiento con otros algoritmos y los resultados para saber si su aplicación permite realizar el estudio de tipos de datos complejos.

Una parte importante del estudio fue la escogencia del conjunto de datos. Se buscaron en diversas fuentes conjuntos de datos que contengan estructuras datos complejas como secuencias de ADN, estructuras químicas o representación de datos codificados. En algunos casos, se encontró dificultad en la comprensión de los datos por su alto contenido técnico en una materia específica. Por ejemplo, un conjunto de datos sobre estudios relacionados a la leucemia requería de un extenso conocimiento en Medicina para ser interpretado, por lo cual fue descartado.
Por lo anterior, los conjuntos de datos que se exponen más adelante son lo suficientemente robustos, completos y comprensibles para que su resultado pueda ser interpretado en este trabajo, sin necesidad de consultar algún experto en el dominio al que se refiere el conjunto de datos. En la revisión previa de los mismos, se escogieron para el estudio los conjuntos de datos de la EPA (Enviromental Protection Agency), pues cumplían con las condiciones necesarias para el estudio contenían tres tipos de datos diferentes (nominales, numéricos y estructurados), su tamaño era los suficientemente robusto para un análisis de minería de datos y los datos, aunque complejos son comprensibles. A continuación aparece la descripción detallada de cada conjunto de datos.

4.1 Descripción de los Conjuntos de Datos

Estos conjuntos de datos han sido tomados de Computational Toxicology Research Program (22). Específicamente, del programa *DSSTox*, este proyecto liderado por la EPA (Enviromental Protection Agency), tiene como finalidad crear un repositorio de datos para mejorar y predecir las capacidades toxicológicas de diferentes sustancias. Los objetivos de proyecto son:

- Fomentar el uso de campos de estructura química estándar (DSSTox) tar el uso de campos de estructura química estándar (DSSTox) de datos normalizado y estructura de archivos de formato SDF para la publicación de datos sobre la toxicidad química

- Aplicar estrictamente los Procedimientos de Revisión de Calidad del *DSSTox* para la notación de las estructuras de información química y químicos asociados con los datos de toxicidad.

- Coordinar con el exterior los esfuerzos públicos para fomentar la anotación de la estructura química, estandarización de los datos y el libre acceso del público a los datos de toxicidad de los archivos.

- Involucrar a la comunidad de usuarios en el esfuerzo para migrar los datos de toxicidad y las listas de toxicidad relacionadas con el producto químico en el formato estándar para la publicación de DSSTox.

- Proporcionar un acceso abierto a los datos de toxicidad para los archivos de la estructura químico-analógica de búsqueda y facilitar el desarrollo de modelos mejorados para predicción de la toxicidad basado en la estructura química.

- Coordinar el desarrollo del proyecto con las metas y objetivos del Programa de Toxicología Computacional de la EPA para mejorar la capacidad de los científicos de la EPA y los reguladores de integrar, analizar y utilizar datos de toxicidad química desde una perspectiva estructural.

A continuación, aparece una explicación de cada uno de los conjuntos de datos involucrados en los experimentos

4.1.1 Conjunto de Datos DBPCAN:

El conjunto de Datos DBPCAN, es producto de la publicación de (23), que contiene las estimaciones de potencial cancerígeno de 209 químicos detectados en muestras de agua potable. El objetivo de este, fue el de proporcionar estimaciones acerca de los carcinógenos para ser utilizado como un factor en la clasificación y priorización de la futura vigilancia de pruebas, y necesidades de investigación en el área del agua potable. Una característica importante según los expertos, es el mecanismo de enfoque basado en la RAS que permite la identificación de estructuras análogas cercanas es la de los grupos químicos funcionales iguales o similares, por los cuales existen datos de genotoxicidad o de carcinogénesis.

Este enfoque, ha sido utilizado eficazmente por la EPA durante muchos años para la detección productos químicos con pocos datos existentes en el marco del programa de notificación de la ley de control de sustancias tóxicas. El DBP (*disinfection by-product chemicals*), se clasifica según una escala semicuantitativa de alto rango para los niveles de la posible carcinogenicidad: Alta(H), de alta y moderada (HM), moderada (M), baja-moderada (LM), bajo (L)

La base de datos DBPCAN lista las 209 sustancias químicas y las estructuras, con su clase química correspondiente, el nivel de potencial carcinogénico de interés (**ActivityConc_Carcinogenicity**). La Información básica de identificación de esta estructura primaria se presenta en los campos: **STRUCTURE_Formula,STRUCTURE_SMILES , STRUCTURE_InChIKey**

Descripción	Cantidad
Atributos	10
Instancias:	172
Distribución de la Clase:	
Low-Moderate	53 (30%)
Marginal	26 (15%)
Low	74 (43%)
Moderate	14 (8.1%)
High-Moderate	3
Marginal(oral)High-Moderate(inhaled)	1
Marginal(oral)High(inhaled)	1

Tabla 4 Información sobre el conjunto de Datos DBPCAN

Nombre del Atributo	Tipo de Dato	Dato
STRUCTURE_□Formula	Cadena	C3H6O
STRUCTURE_MolecularWeight	Numérico	58
STRUCTURE_SMILES	Cadena	CCC=O
STRUCTURE_Parent_SMILES	Cadena	CCC=O
STRUCTURE_InChIKey	Cadena	NBBJYMSMWIIQGU-UHFFFAOYAZ
ActivityScore _DBPCAN	Numérico	30
ActivityConcernLevel _Carcinogenicity	Cadena	Low-Moderate
ActivityConcern Level _RationaleSource	Cadena	Authorcommunication

Tabla 5 Ejemplo del conjunto de Datos DBPCAN

En algunos de los atributos de tipo cadena aparece el acrónimo SMILES (Sistema simplificado de entrada molecular). Se trata de una notación química simplificada que permite representar una estructura química de dos dimensiones en forma lineal de tipo texto, para facilitar la comprensión por una aplicación informática. El código SMILES (24), según el uso de la norma actual genera una estructura molecular única, pero lo inverso no es cierto, es decir, una estructura molecular puede ser representada por múltiples cadenas de código SMILES.

La notación SMILES se emplea comúnmente en la QSAR (Quantitative Structure-Activity Relationship). Se utiliza también en algunos programas comerciales y públicos para la predicción de las propiedades químicas y actividades biológicas.

Además, aparece otra codificación llamada InChiKey, esta fue presentada en septiembre de 2007. Se trata de una longitud fija (25 caracteres) que proporciona una representación condensada digital del identificador InChI que se puede utilizar para facilitar la búsqueda por estructura molecular.

Sustancia	Codificación SMILES :
Etanol	CCO
Ácido acético	CC (= O) O
Ciclohexano	C1CCCCC1
Cloruro de sodio	[Na +]. [Cl-]

Tabla 6 Algunos ejemplos de codificación SMILES

La representación InChi tiene características como:

- Facilitar la búsqueda web, ya complicada por la rotura imprevisible de las cadenas de caracteres InChI en los motores de búsqueda
- Permitir el desarrollo de un servicio de búsqueda basado en la web InChI.
- Permitir una representación InChI que se almacene en campos de longitud fija
- Que la estructura química sea de fácil indexación.
- Permitir la verificación de las cadenas de InChI después de la transmisión por la red.

A continuación se presenta un Ejemplo de InChI con su equivalente InChIKey:

La cafeína:

InChI = 1/C8H10N4O2/c1-10-4-9-6-5 (10) 7 (13) 12 (3) 8 (14) 11 (6) 2/h4H,1-3H3.

InChIKey = RYYVLZVUVIJVGH-UHFFFAOYAW

Explicación:

- Primer bloque (14 letras), codifica esqueleto molecular (conectividad): RYYVLZVUVIJVGH
- Segundo bloque (8 letras), codifica las posiciones de protones (tautómeros), estereoquímica, isótopos, la capa de conectar: UHFFFAOY
- Carácter de la bandera, indica InChI versión, la presencia / ausencia de la capa fija H, isótopos, y estereoquímica: A
- Carácter de comprobación: W.

En los conjuntos de datos estas estructuras son de vital importancia en este trabajo para poder realizar el ejercicio de comparar cadenas usando la distancia de edición.

4.1.2 Conjunto de Datos ISSCAN

Esta base de datos se origina en la experiencia de múltiples investigadores del Departamento de Medio Ambiente y Prevención Primaria, en el campo de las relaciones estructura_actividad (SAR). Estas investigaciones buscan desarrollar modelos que teóricamente puedan predecir el nivel cancerígeno de las sustancias químicas. Los productos químicos han sido el tema de estudio para la clasificación de carcinogenicidad por varias agencias y organismos científicos.

Estas bases de datos han sido diseñadas específicamente como una herramienta de apoyo a la toma de decisiones para guiar la aplicación del enfoque SAR. Algunas de las agencias colaboradoras de este estudio son:

- CPDB (Berkeley Carcinogenic Potency DataBase);
- TOXNET CCRIS (database CCRIS from the cluster of toxicological databases TOXNET);
- NTP (National Toxicology Program; the Technical Report number is also provided);
- IARC (International Agency for Research on Cancer);
- EINECS (European Inventory of Existing Commercial Chemical Substances).

Descripción	Cantidad
Instancias	1136
Atributos	14
Distribución de la Clase:	
3.0	(709) 62%
2.0	(68) 5.95%
1.0	(359) 31 %

Tabla 7 Información sobre el conjunto de datos ISSCAN.

Los atributos de especial interés son:

Nombre del Atributo	Tipo	Ejemplo
ChemName	Nominal	4-Dimethylaminoazobenzene
MolWeight	Numérico	225
Formula	Cadena	C14H15N3
SMILES	Cadena	CN(C1=CC=C(C=C1)N=N/C2=CC=CC=C2)C
SAL	Nominal	3.0
Rat_Male_Canc	Nominal	ND
Rat_Female_Canc	Nominal	3.0
Mouse_Male_Canc	Nominal	3.0
Mouse_Female_Canc	Nominal	1.0
Rat_Male_NTP	Nominal	ND
Rat_Female_NTP	Nominal	ND
Mouse_Male_NTP	Nominal	ND
Mouse_Female_NTP	Nominal	ND
Canc	Numérico	1.0

Tabla 8 Ejemplo del conjunto de datos ISSCAN.

También, aparecen estructuras de tipo InChiKey, SMILES que son atributos de tipo cadena. La descripción de los atributos tipo InChiKey, SMILES como aparecen en la tabla 7.

4.1.3 Conjunto de datos FDAMDD

La base de datos del Centro de Evaluación e Investigación de Drogas, la Oficina de Farmacia y el Staff de Análisis de Seguridad Informática y Computación de la Dosis Diaria recomendada (FDAMDD) (22) contiene los valores de más de 1200 productos farmacéuticos. La mayor parte de los valores corresponde a la dosis diaria máxima recomendada (MRDD) en la base de datos se determina a partir de los ensayos clínicos farmacéuticos que emplean una vía oral de exposición y tratamientos al día, generalmente en periodos de 3 a 12 meses. Es de destacar que algunos fármacos tienen diferentes valores MRDD para hombres y mujeres adultos, niños o pacientes de edad avanzada.

Para este conjunto de datos se han agrupado los atributos para ayudar en la clasificación, como clases artificiales ya que no son representativos en el conjunto de datos, debido a que el número de instancias por clase es muy reducido.

Descripción	Cantidad
Instancias:	1093
Atributos:	10
Distribución de la clase:	
Numero de ítems únicos:	296
Antinflamatorio	(19) 1.73%
Antidiabético	(15) 1.32%
Antibacterial	(124) 11.31%
Antineoplásico	(49) 4.4%
Antihistamínico	(22) 2.1%
Sedativo-hipnótico	(23) 2.13%
Antidepresivo	(24) 2.13%
Antihipertensivo	(38) 3.4%

Tabla 9 Información del conjunto de datos FDAMDD

Algunos de los atributos de tipo estructurado, que aparecen en este conjunto de datos se han explicado previamente, son los que contienen estructuras de tipo SMILES o Inchi Key. En particular aparecen atributos como ActivitySco-re_FDAMDD, este mide la potencia del medicamento administrado en una dosis diaria. ActivityScore FDAMDD usa el atributo anterior para crear una categoría de la potencia del medicamento.

Nombre del atributo	Tipo	Ejemplo
STRUCTURE_□Formula	Cadena	C13H18O2
STRUCTURE_ Molecular_Weight	Número	206
STRUCTURE_SMILES	Cadena	CC(C1=CC=C(C=C1)CC(C)C)C(=O)O
STRUCTURE_InChIKey	Cadena	HEFNNWSXXWATRW-YHMJCDSICY
ActivityScore_FDAMDD	Número	19
ActivityCategory_ MRDD_mmol	Nominal	Low
ActivityCategory_ MCASE_mg	Nominal	Low
Therapeutic Category	Nominal	art1

Tabla 10 Información del conjunto de datos FDAMDD

4.2 Experimentación.

Con el fin de realizar las pruebas del DBDT usando datos estructurados del tipo cadenas, se realizó la modificación de la matriz de distancia para calcular el valor de los atributos de acuerdo a su tipo. El DBDT original realiza el cálculo de la matriz de distancia para atributos numéricos y nominales, para introducir atributos estructurados como cadenas, se modificó el para calcular la distancia de los atributos de tipo no estructurado, usando la distancia de edición. Se realizaron pruebas con el DBDT y los conjuntos de datos mencionados anteriormente. El siguiente paso, fue el de comparar con otros algoritmos similares, para esto se escogió el algoritmo KNN y se modificó su versión original para manejar atributos del tipo cadena usando también la distancia de edición. La realización del experimento se ha llevado a cabo con 3 datasets y se ha realizado una validación cruzada de 10 x 10 para cada dataset y método. Adicionalmente se utilizó un método conocido como *Smote* (25), para dar balance a las clases del dataset y garantizar que las clases están igualmente representadas en el dataset. Como se ha mencionado en el capítulo dos de este documento, los algoritmos de minería de datos pueden tener una o más funciones de evaluación. Las funciones de evaluación que se usaron en este trabajo son:

- La precisión media es decir el ratio de aciertos (número de aciertos divido entre número de fallas), este método se utiliza como medida de error cualitativa.

- El AUC (Area under the ROC Curve), esta función de evaluación está basada en el análisis ROC (11 pág. 469), -usualmente es usado como medida de calidad en el ranking- que permite seleccionar de un conjunto de clasificadores el que mejor se comporte en una serie de circunstancias determinadas a posteriori. Una vez realizado el análisis ROC de cada clasificador, se puede realizar el análisis AUC, que permite a través de una estimación de probabilidad conocer cuál será el clasificador más óptimo. Una explicación más detallada se puede ver en el Apéndice B.

Además, en la ejecución de los experimentos utilizando el DBDT con datos estructurados se permite calcular los datos para atributos nominales, numéricos y estructurados (secuencias). EL DBDT también es flexible al permitir clasificar por proximidad -clasificación de instancias basadas en la distancia- o por densidad -clasificación de la instancia basada en la densidad-.

En cuanto a las particiones de los datos el DBDT utiliza los siguientes criterios:

- Gain Ratio(GR)
- Information Gain(IG)

En el experimento, también es preciso aclarar que las pruebas incluyen los experimentos con ambos criterios de partición, estos criterios están explicados detalladamente en el apartado que habla sobre arboles de decisión. Además, de los criterios de partición se realizó una validación cruzada de los datos de 10 X 10 para hacer la pruebas más aleatorias.

Algunas aclaraciones previas a los resultados se presentan a continuación:

1. Al realizar la validación para los atributos con datos estructurados, se reveló que al comparar dos instancias existen tres casos:

   - El primero, en que el valor es cero por que ambos atributos son iguales por ejemplo si la cadena NWQWQKUXRJYXFH-UHFFFAOYAA se evalúa contra sí misma la distancia entre la dos será igual a cero.

   - El segundo, donde el valor es la distancia de edición entre las dos cadenas. Por ejemplo, una cadena (A) <- NWQWQKUXRJYXFH-UHFFFAOYAA y otra cadena (B) <-NWQWQKUXRJYXFH-UHFFFAOY2 , si se aplica la distancia de edición, su resultado será igual a 2, ya que es el mínimo número de operaciones necesarias para convertir la cadena A en B.

   - Un tercero, donde la instancia no está dentro del conjunto de pruebas. Es decir si existe una cadena NWQWQKUXRJYXFHUHFFFAOYAA y un subconjunto de pruebas A = {ABBCCDD, ACCDDBB,ABBCCDD} . Como se puede ver en este subconjunto escogido al azar no existe la cadena antes mencionada con el fin, de realizar una medición más precisa se decidió para este caso introducir el promedio de las distancias.

2. En algunos casos, el AUC daba como resultado NAN (Not a number). En el conjunto de datos FDAMDD, esto ocurre porque algunos de los atributos tenían muy pocos ejemplos e inducían al error, lo cual no permitían un cálculo del Área bajo la Curva (AUC) acertado, como consecuencia se han agrupado atributos para conseguir un valor adecuado para el AUC. Esta agrupación se realizó debido a que el número de instancias era muy pequeño y genera errores en el conjunto de datos que no era representativo. Por ejemplo en el conjunto de datos FDAMDD3 se encontró que algunas de las clases tenían solo entre 5 o 15 ejemplos lo cual generaba problemas de clasificación.

3. Los algoritmos clásicos de clasificación KNN y C4.5, manejan las particiones de datos nominales y numéricos, no tienen en cuenta los datos estructurados pues estos son usados como atributos nominales, esto es importante porque solo hace una clasificación binaria omitiendo las relaciones que puedan existir entre datos estructurados similares.

A continuación, aparecen los experimentos incluyendo los algoritmos arboles de decisión y conjunto reglas arboles de decisión derivados de cada uno

4.2.1 Dataset: DBPCAN(DBPCANnoStructures42)

En el conjunto de datos DBPCAN se realizaron algunos cambios respecto a la información original con el fin de dar mayor consistencia a la prueba. El cambio principal se realizó al reducir el número de clases, pues la cantidad de ejemplos por clase hacia que el resultado de las pruebas no fueran relevantes para el estudio. El número de clases se redujo a tres "low", "low- moderate" y "moderate", no fueron modificadas; En la clase "Low-moderate" se agruparon las clases: "marginal moderate", "Marginal (oral)", "High-Moderate (inhaled)", "Marginal (oral)" "High(inhaled)", "High moderate".Para evaluar los datos estadísticos y garantizar la independencia de las pruebas se realizó una validación cruzada de 10 x10.

Etiqueta	Tamaño	Tamaño(SMOTE)
Distribución de la clase:		
Low_moderate	53	110
Low	74	116
Moderate	44	98
TOTAL	209	234

Tabla 11 Informativa DBPCAN

				Clasificado por:			
				Densidad/ GR	**Densidad/ IG**	**Prox / GR**	**Prox/ IG**
Alg.	**KNN**	**KNN-S***	**C4.5**	**DBDT (ch.=2)**	**DBDT (ch.=2)**	**DBDT (ch.=2)**	**DBDT (ch.=2)**
Prec. Media	99.6941 ± 0.9223	99.69412 ± 0.92233	91.7026 ± 0.097	97.5132 ± 6.88045	97.48200 ± 6.87613	0.9653 ± 0.0127	99.96875 ± 0.31249
AUC	0.99616 ± 0.0017	0.995157 ± 0.01567	0.95199 ± 0.0126	0.95199672 ± 0.01262177	0.95173 ± 0.01276	0.95199 ± 0.01262	0.951733 ± 0.012761

Tabla 12 Resultados DBPCAN
***KNN-S para datos estructurados usando la distancia de edición**

Para este conjunto de datos, el árbol de decisión correspondiente permite deducir algunas de las siguientes reglas

1. SI Activity Outcome =2 entonces Є CLASE *Low*
2. Si Activity Outcome es =0 y InChiKey Є { NWQWQKUXRJYXFH-UHFFFAOYAA} y InchiKey Є { SFWAQPWRFZOPKA-UHFFFAOYAV} ActivityScore_DBPCAN=30 entonces Є CLASE *Low-moderate*
3. SI Activity Outcome es =0 y InChiKey Є { NWQWQKUXRJYXFH-UHFFFAOYAA} y InchiKey Є {SFWAQPWRFZOPKA-UHFFFAOYAV} ActivityScore_DBPCAN=70 entonces Є CLASE *Moderate*
4. SI Activity Outcome es =0 y InChiKey Є set{ NWQWQKUXRJYXFH- UHFFFAOYAZ} y ActivityScore_DBPCAN=70 entonces Є CLASE *Moderat*

En la siguiente página se puede ver el árbol de decisión generado por el DBDT.

Ilustración 7 Árbol de decisión DBPCAN hasta el 4º nivel

4.2.2 Dataset: ISSCAN(ISSCANv2)

En el Conjunto de datos ISSCAN se realiza un agrupamiento de clase, se hace para las clases 1 y 2, ya que la clase 2 tiene muy pocas instancias. Se realiza este ajuste para evitar valores anómalos.

Etiqueta	Tamaño	Tamaño (Smote)
Distribución de la clase:		
3	748	748
1	388	766
TOTAL	1136	1524

Tabla 13 Informativa ISSCAN

				Clasificado por:			
				Densidad/ GR	***Densidad / IG***	***Prox/ GR***	***Prox/ IG***
Alg.	***KNN***	***KNN-S****	***C4.5***	***DBDT (ch.=2)***	***DBDT (ch.=2)***	***DBDT (ch.=2)***	***DBDT (ch.=2)***
Prec. Media	94.6460 ± 1.9091	95.45368 ± 1.569286	96.3716 ± 1.6945	97.56566 ± 1.24210	97.55250 ± 1.23341	96.7243 ± 1.6711	96.6277 ± 1.7701
AUC	0.996006 ± 0.003384	0.9932 ± 0.0048	0.99600 ± 0.0033	0.9693302 ± 0.01264	0.969302 ± 0.0120	0.9530 ± 0.0198	0.9520 ± 0.0201

Tabla 14 Resultados ISSCAN

***KNN-S para datos estructurados usando la distancia de edición**

Algunas de las reglas que se pueden deducir del árbol correspondiente son:

1. Si Rat_Male_Canc=1 entonces ∈ CLASE *3*
2. Si Rat_Male_Canc=2 y Mouse_Female_Canc=2 entonces ∈ CLASE *3*
3. Si Rat_Male_Canc=2 y Mouse_Female_Canc=0 y Rat_female_canc=0 y Mouse_Female_Canc=0 y SMILES CN(C1=CC=C(C=C1)N=N/C2=CC=CC=C2)C} y ChemName=0 entonces ∈ CLASE *1*
4. Si Rat_Male_Canc=2 y Mouse_Female_Canc=2 y Rat_female_canc=0 Mouse_Female_Canc=0 y SMILES set ∈ {CN(C1=CC=C(C=C1)N=N/C2=CC=CC=C2)C} y ChemName=636 entonces ∈ CLASE *1*

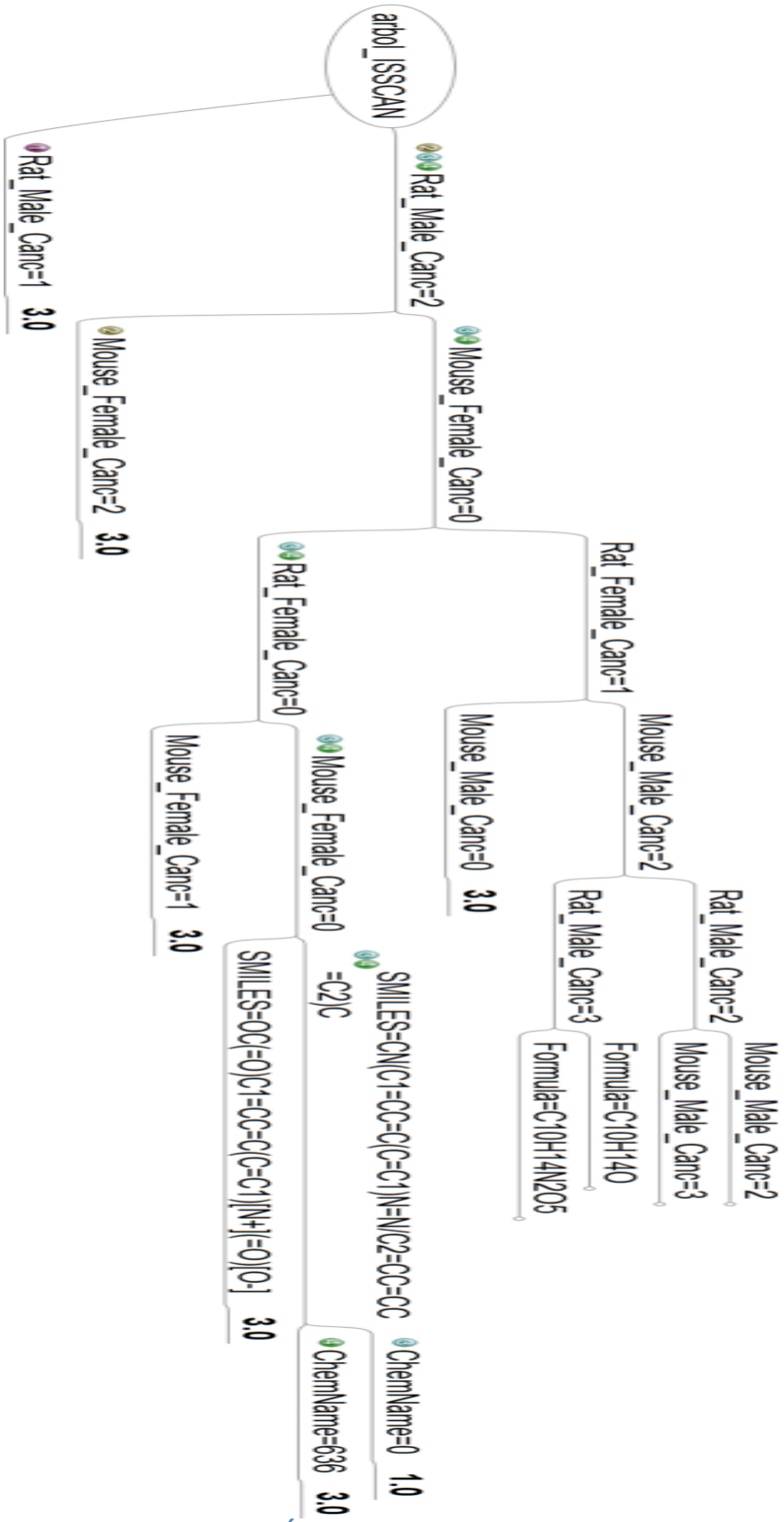


Ilustración 8 Árbol de decisión ISSCAN hasta el 6º nivel

4.2.3 Dataset: FDAMDD-3

Para este dataset se realizó una agrupación de clases donde se creó una clase artificial (art1), que agrupa las clases que contienen de 1 a 15 instancias.

Etiqueta	Tamaño	Tamaño (Smote)
Distribución de la clase:		
art1	145	145
Antibacterial	124	124
diagnostic aid	148	148
Antineoplastic	92	184
TOTAL	509	601

Tabla 15 Informativa del FDAMDD-3

				Clasificado por:			
				Densidad /GR	**Densidad / IG**	**Prox. /GR**	**Prox. /IG**
Alg.	**KNN**	**KNN-S**	**C4.5**	**DBDT (ch.=2)**	**DBDT (ch.=2)**	**DBDT (ch=2)**	**DBDT (ch.=2)**
Prec. Media	47.0184 ± 6.4962	30.92549 ± 5.8280	47.9572 ± 6.8963	52.3572 ± 6.8954	51.6129 ± 7.8802	51.9474 ± 7.6939	51.5735 ± 7.8932
AUC	0.7098 ± 0. 0806	0.5890 ± 0.1090	0.6488 ± 0.1011	0.5929 ± 0.0979	0.5842 ± 0.0834	0.5911 ± 0.0888	0.5843 ± 0.0834

Tabla 16 Resultados FDAMDD-3

***KNN-S para datos estructurados usando la distancia de edición**

Algunas de las reglas que se pueden deducir del árbol correspondiente son:

1. SI ActivityCategory_MCASE_mg=1.0 y Peso molecular=604 Activityscore FDAMDD = 55 y Structure_FORMULA=471 Є CLASE *antineoplastic.*
2. SI ActivityCategory_MCASE_mg=2.0 y Activityscore_FDAMDD = 23 y Peso molecular=843 y Structure_FORMULA=107 Є CLASE *antibacterial*.
3. SI ActivityCategory_MCASE_mg=2.0 y Activityscore_FDAMDD = 23 y el Peso molecular=843 y Structure_FORMULA=192 Є CLASE *diagnostic_aidl*.
4. SI ActivityCategory_MCASE_mg=1.0 y Peso molecular=604y Activityscore FDAMDD = 54 y Structure_FORMULA=181 Є CLASE *diagnostic_aid*.

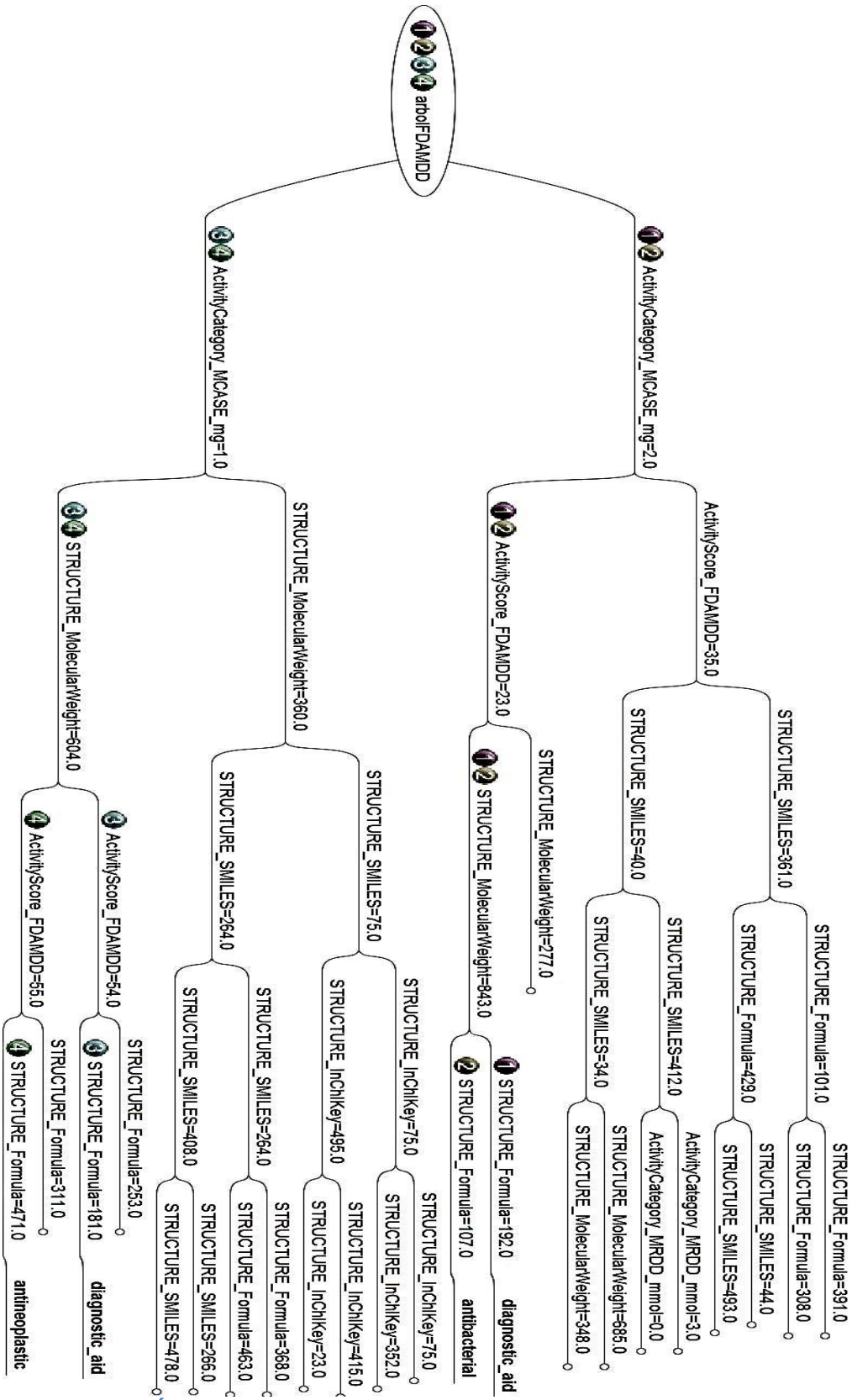


Ilustración 9 Árbol FDAMDD hasta el 5º nivel

Se realizó también un análisis adicional para evaluar si las diferencias encontradas en estas pruebas son significativas, el método escogido es el test estadístico de t-student. Para analizar si las diferencias son significativas se utilizó un nivel de confianza de ∞= 0.$\propto = 0.05$ y N= 64, las diferencias significativas se resaltan en negrita. En la Tabla 18, se muestran los resultados de por dataset, algoritmo y método de evaluación, indicando con W cuando el DBDT gana, T cuando empata y L cuando pierde. Delas tablas se puede concluir que el DBDT tiene un comportamiento aceptable respecto a los algoritmos. En la precisión, le DBDT se destaca por encima de los otros datasets. En el AUC se muestra el buen rendimiento del indicador aunque muestra perdidas en el dataset ISSCAN. Los resultados obtenidos para cada conjunto de datos aparecen a continuación:

Dataset	DBDT		KNN-S		KNN		C 4.5	
	Precisión	AUC	Precisión	AUC	Precisión	AUC	Precisión	AUC
DBPCAN	**0.99950**	0.950414	0.96505	0.967731	0.99608	**0.995477**	0.993484	0.950825
FDDAMD	**0.51730**	**0.797786**	0.45272	0.708328	0.46827	0.708797	0.46642	0.655966
ISSCAN	**0.97462**	0.968781	0.95436	**0.995730**	0.95772	0.995181	0.97455	0.984686

Tabla 17 Comparación de Medias entre DBDT, KNN-S, KNN, C4.5

Dataset	DBDT Vs. KNN-S		DBDT Vs. KNN		DBDT Vs.C 4.5	
	Precisión	AUC	Precisión	AUC	Precisión	AUC
DBPCAN	W	W	W	T	W	T
FDDAMD	W	W	W	W	W	W
ISSCAN	W	L	W	L	W	L

Tabla 18 Resultados usando t-Student.

5 Conclusiones

5.1 Conclusiones

Ha sido el objetivo de este trabajo, realizar una comparación del algoritmo derivado del DBDT, para poder analizar datos estructurados. Se han utilizado algoritmos como el j48, C4.5 y el KNN y KNN-S una versión modificada para este mismo ejercicio con el fin de comparar datos estructurados.

La experimentación realizada sobre los datos estructurados ha abarcado el uso de la distancia de edición, aplicándola con el algoritmo DBDT. Esto ha permitido el análisis de estructuras de datos de una manera similar a otros conjuntos de datos teniendo en cuenta la naturaleza de los conjuntos de datos estructurados. Los resultados como lo muestran los experimentos han sido similares y en algunos casos mejores a los realizados con otros algoritmos que tratan los datos estructurados como atributos nominales, este uso de la distancia de edición es muy importante porque además permite realizar una aproximación de diferente al manejo de datos estructurados, ya que los otros algoritmos no incluyen la relación que existe entre tipos de datos estructurados y la diferencia a través de la comparación basada en la distancia.

Lo anterior permite demostrar que el uso de la estructura es provechoso en operadores para clasificar estos tipos de datos utilizando esta técnica. Así lo demuestra además los test estadísticos realizados del DBDT vs los otros algoritmos, el DBDT se muestra como un clasificador aceptable y competitivo respecto a los clasificadores clásicos. La flexibilidad de poder manejar varios tipos de datos a clasificar, es un valor adicional dado que las aproximaciones clásicas están limitadas a la exploración de tipos de datos numéricos y nominales. La aplicación del uso de distancia para diferentes tipos de datos hace que sea una forma sintética, que se puede aplicar a diversos tipos de datos como series, listas, conjuntos, grafos e incluso textos, si es posible conseguir una fórmula de distancia adecuada. También es importante mencionar que la aproximación del DBDT realiza un pre-cálculo de las distancias, lo hace más eficiente en el procesamiento que los otros clasificadores lo cual puede significar una ventaja por su costo en el manejo de grandes volúmenes de datos. También, se muestra en este trabajo, la dificultad en la comprensibilidad de los datos estructurados aunque se usen arboles de decisión, no es la representación más óptima.

A través de la experimentación se puede mencionar algunas aplicaciones del trabajo a futuro, como son el de poder realizar una clasificación con otros tipos de datos estructurados como por ejemplo la clasificación de subconjuntos de datos estructurados entre los conjuntos previamente mostrados. La mejora del DBDT para el manejo de los datos estructurados y su aplicación a diversos tipos de datos y la relación entre las subcadenas similares sobre el mismo conjunto

de datos. También el estudio de técnicas de visualización e interpretación del conocimiento adquirido pues como se ve en los arboles no es exactamente sencillo de interpretar. Por último, la comparación con otras técnicas, si existen que tratan datos estructurados o la comparación del mismo frente a otros tipos de datos (arboles, grafos, conjuntos).

5.2 Trabajo a futuro

Se pueden pensar diversas variantes de esta investigación. La primera, puede ser en aumentar la comprensibilidad de los arboles decisión, ya que se percibe que no es sencillo para el usuario interpretar extender a diferentes tipos de datos. Además se puede investigar la aplicación a otros tipos de datos que permitan ampliar el alcance del algoritmo a otros tipos de datos más complejos como secuencias, conjuntos o grafos y aplicar también la investigación con conjuntos de datos más robustos. También es posible contemplar la aplicación del DBDT a otros métodos de aprendizaje y como el clustering, la regresión u otros métodos de aprendizaje.

6 Bibliografía

1. **Hand, David Mannila, Heikki and Smyth, Padhraic.** Introduction to Datamining. *Principles of datamining.* s.l. : The MIT Press, 2001.
2. **Quinlan, Ross.** *Induction of decision trees p 81-106.* 1983.
3. **Pazzani, M. & Billsus, D.** Learning and Revising User Profiles: The identification of interesting web sites, 313-331. 1997.
4. **Howe, Dreilinger.** SavvySearch: A Meta-Search Engine that Learns which Search Engines to Query. 1996.
5. **Knight, Koehn.** Learning a Translation Lexicon from Monolingual Corpora. 2002.
6. **Frank, Ian, Witten W. & Eibe.** Datamining: Practical Machine Learning Tools and techniques. s.l. : elsevier, 2002.
7. **Estruch, Vicent.** Bridging the Gap between Distance and generalisation: symbolic learning in metricspace. Valencia : s.n., 2008.
8. **Frank, Ian, Witten W. & Eibe.** Data Mining: Practical Machine Learning Tools and Techniques. [aut. libro] Witten Frank. *Data Mining: Practical Machine Learning Tools and Techniques.* 2011.
9. **Hand, Manilla, Smyth.** *Principles of Datamining.* s.l. : MIT press, 2001.
10. **Russel, Stuart y Norvig, Peter.** Inteligencia artificial. un enfoque moderno,Capitulo 18. s.l. : Pearson, 2004.
11. **Hernández Orallo, José Quintana,M.José Ramírez Ferri Ramírez, Cèsar.** *Introduccion a la Mineria de Datos.* Madrid : Pearson, 2004.
12. **Mitchell, Tom.** *Machine Learning- Ch3.* s.l. : Mc Graw Hill, 1998.
13. **Breiman, Leo y Friedman, J. H., Olshen, R. A., & Stone, C. J.** *Classification and regression trees.* 1984.
14. **Michalsk J.i & Larson, L.S.** *Incremental generation of VLi hypoteses: the underlying methodology and the description of the program AQ11.* 1983.
15. **Shafer, John Rakesh,Agrawal Mehta, Manish.** SPRINT: A Scalable Parallel Classifier for Data Mining. 1996.
16. **Estruch, Vicent.** Distance based generalisation operators. Tesis Doctoral DSIC. *Bridging the Gap between distance and generalisation: syimbolic learning metric spaces.* Valencia : s.n., 2008.
17. **Ramon J., Bruynooghe M., and Van Laer. W. In.** Distance measures between atoms. [aut. libro] CompulogNet Area Meeting on Computational Logic and Machine Learing. 1998.
18. **Mitchell, Tom.** *Machine Learning.* Bogota : Mac Graw Hill, 1997.
19. **Jr, E. Fix and J.L. Hodges.** Discrminatory anlaysis, non parametric discrimination,. s.l. : USAF school of aviation Medicine, randolf field, 1951.
20. **Jones, J. Quinlan, R. and Cameron R.M.** Foil: A midterm report. s.l. : Springer 2003, In Proc. of the European, Vol. 3.
21. **Breiman L., Friedman J.H., Olshen R.A., and Stone C.J.** Classification and regression trees. s.l. : Chapman and Hall, 1984.
22. EPA. [aut. libro] Enviromental Protection Agency. *http://www.epa.gov/* . 2007.

23. **Woo, Y.T., C.R. Williams, N. Fields, and A.M. Richard.** *DSSTox EPA Water Disinfection By-Products with Carcinogenicity Estimates Database (DBPCAN): SDF Files and Documentation.* 2008.
24. **Daylight.** http://www.daylight.com/. 2008.
25. **Nitesh V. Chawla, Kevin W. Bowyer et all.** *SMOTE: Synthetic Minority Over-sampling Technique.* 2002. Vols. 321-357.
26. **Araujo, Basilio Sierra.** *aprendizaje automático: Conceptos básicos y avanzados.* madrid : Pearson Educacion, 2006.

Apéndices

Apéndice A. Varios.

A1.Código. Algoritmo para el Cálculo de la Distancia de Edición

```
Entrada: cad0 es una cadena de longitud lon1, cad1 es una cadena de lon2
Salida: Matriz d[i, j];
Inicio
Para  i =0 …..lon1 haga
    d[i, j]=i
Fin
Para  i =0 …..lon2 haga
    d[0, j]=j
Fin
Para i=1 …. Lon1
 Para j =1……. Lon2
 Si cad0 [i]=cad1[j] entonces
 Costo=0;
sino
 Costo=1;
        d[i, j] := mínimo(
                    d[i-1, j] + 1,     // Borrado
                    d[i, j-1] + 1,     // Inserción
                    d[i-1, j-1] + cost   //Sustitución
                  )
 Fin
Fin

 Devolver d[lon1, lon2]
```

Apéndice B. Análisis AUC.

Para entender el análisis AUC es preciso mencionar el análisis ROC pues este depende del AUC. El análisis ROC según (11 pág. 468), permite la selección del clasificador que tiene un comportamiento óptimo y hace que la evaluación de los clasificadores para minería de datos sea más independiente y completa que el análisis usando la medida de precisión. El análisis ROC funciona normalmente para problemas de dos clases (positiva y negativa) y se utiliza un matriz de confusión para representarlo.

	Real	
Estimado	True positives (TP)	False Positive (FN)
	False Negative (FN)	True Negative (TN)

Tabla 19.Matriz de confusión análisis ROC

Estos valores son normalizados así:
True Positives Rate (TPR) = TP/(TP+FN)
False Positives Rate (FPR) = TN/(FP+TN)
False Negative Rate (FNR) = FN/(TP+FN)
True Negative Rate (TNR) = TN/(FP+TN)

Al construir una matriz de confusión normalizada se verá así:

	Real	
Estimado	FPR	FPR
	FNR	TNR

Tabla 20 matriz de confusión normalizada

Donde TPR=1-FNR y FPR=1-TNR.

Una vez se ha realizado este análisis es posible construir un espacio bidimensional (Espacio ROC) que permite la visualización de los clasificadores. Este espacio utiliza el valor TPR para el eje Y y FPR para los valores de X. La interpretación del análisis ROC se puede dar con una matriz de confusión así:

	Real	
Estimado	30	40
	20	70

A partir de la matriz de confusión se calculan los valores TPR=0.75 y FPR=0.5, si se grafica en un espacio bidimensional el resultado será el siguiente

Al representar un clasificador en el espacio ROC se cumple una propiedad fundamental para el análisis de datos, como se cita en (11 pág. 470) :

- *Dado un conjunto de clasificadores se pueden descartar aquellos que no caigan en la superficie convexa formada por todos los puntos de clasificadores como mayor área bajo la superficie convexa o AUC.*

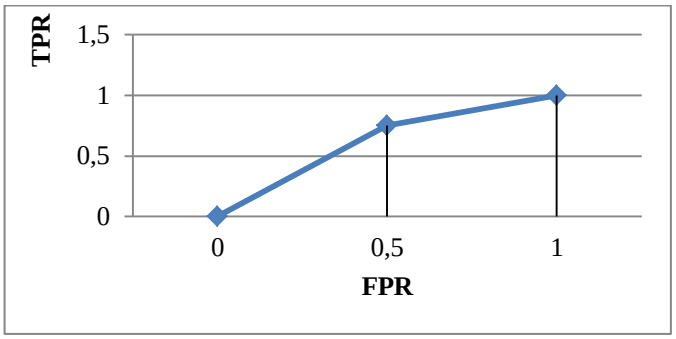


Ilustración 10. Clasificador en el espacio ROC

Por lo tanto para el aprendizaje de los clasificadores cuando son sensibles al coste, el mejor sistema de aprendizaje será aquel que produzca el conjunto de clasificadores con mayor área sobre la superficie convexa o AUC. Para ilustrar mejor el análisis de AUC se presenta el siguiente gráfico:

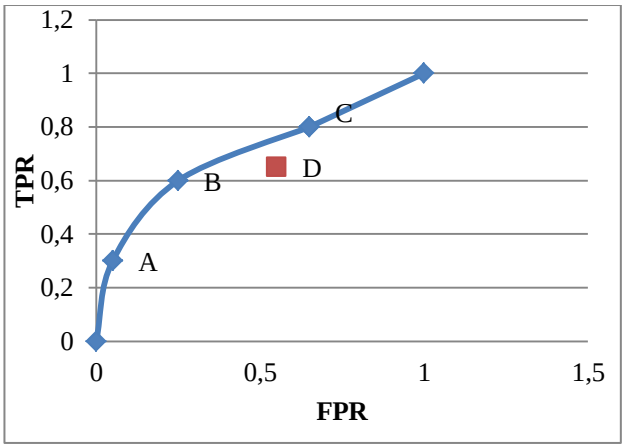


Ilustración 11 Diagrama de ROC

En el ejemplo de la ilustración serán mejores clasificadores A,B y C, el clasificador D será descartado pues se encuentra en el interior de la superficie convexa. Una vez escogidos lo clasificadores óptimos A, B y C, se calcula el valor de los costes de clasificación errónea y la distribución de las clases Así con estos valores se calcula una pendiente M conocida como tendencia, que será dada por :

$$m = \frac{p(N) \cdot C(P|N)}{p(P) \cdot C(N|P)},$$

Donde p(N) es la probabilidad de que un nuevo dato sea de la clase N, C(P|N) es el coste asociado a clasificar un objeto como P cuando es N, p(P) es la probabilidad de que un objeto sea de la clase P,C(N|P) es el coste asociado de clasificar un objeto N cuando es P. El valor de M es la pendiente de una línea recta que atraviesa el punto (0,1), esa recta se puede mover hasta encontrar el primer punto que intersecte la Curva ROC y es el que dará el clasificador más óptimo para la tarea que se ha asignado. Cuando un problema contiene c clases, el espacio ROC tiene c(c-1) dimensiones. Lo anterior hace inviable el cálculo de superficies convexas para c>2, como esto se han estudiado diversas aproximaciones para calcular el AUC de clasificadores múltiples que se citan en (11 pág. 471)

Apéndice C: Código

C1.Código implementación de la distancia de edición para la matriz de distancia.

```
public static int getLevenshteinDistance (String s, String t) {
     if (s == null || t == null) {
       throw new IllegalArgumentException("Strings must not be null");
     }


     int n = s.length(); // length of s
     int m = t.length(); // length of t

     if (n == 0) {
       return m;
     } else if (m == 0) {
       return n;
     }

     int p[] = new int[n+1]; //arreglo previo para el costo
     int d[] = new int[n+1]; // arreglo para el costo
     int _d[];
     // indices
     int i; //
     int j; //
     char t_j; // t

     int cost; // costo

     for (i = 0; i<=n; i++) {
        p[i] = i;
     }

     for (j = 1; j<=m; j++) {
        t_j = t.charAt(j-1);
        d[0] = j;

        for (i=1; i<=n; i++) {
           cost = s.charAt(i-1)==t_j ? 0 : 1;
           //manejo del costo mínimo
           d[i] = Math.min(Math.min(d[i-1]+1, p[i]+1),  p[i-1]+cost);
                }
       // Copia la distancia presente a la fila previa.
       _d = p;
        p = d;
        d = _d;
         }

          return p[n];
      };
```

C2.Código para el KNN-S

Código donde se ejecuta el Algoritmo del KNN-s

```
 public void Checking_KNNMetric(MetricSpace ms, int maxBranches, int clas-
sify_criterion, int splitting_criterion, boolean trace, boolean AUC, bool-
ean Acc, int neighbors,int [] attributes_method)
 {

    Random random;
     int seed; int k;
   double real_class, classified_class,classified_class_knn,right;
   Instances train, test;
    Enumeration enume;
   Instance instance;
    IBkMetric Knn;
   racsUtils rUtils;

   seed=1;

//instanciacion del algoritmo KNN-S
Knn = new IBkMetric(k,ms);

   Knn.m_Original=dataSet1;
//  Configuración de la clase original de la distribución (pre prunning)
   rUtils = new racsUtils(dataSet);
      try
      {
       for (int i_rep = 0; i_rep < rep; i_rep++) {
            /* Se  imprime la información relacionada con la repetición*/

       java.util.Date ahorainicio = new java.util.Date();
         System.out.println("Contador rep inicio"+i_rep+ "  "+ ahorai-
      nicio.toString());

      }
       catch(Exception e)
       {
            System.out.println(e);
       };
   };
```

….. continua en la siguiente pagina

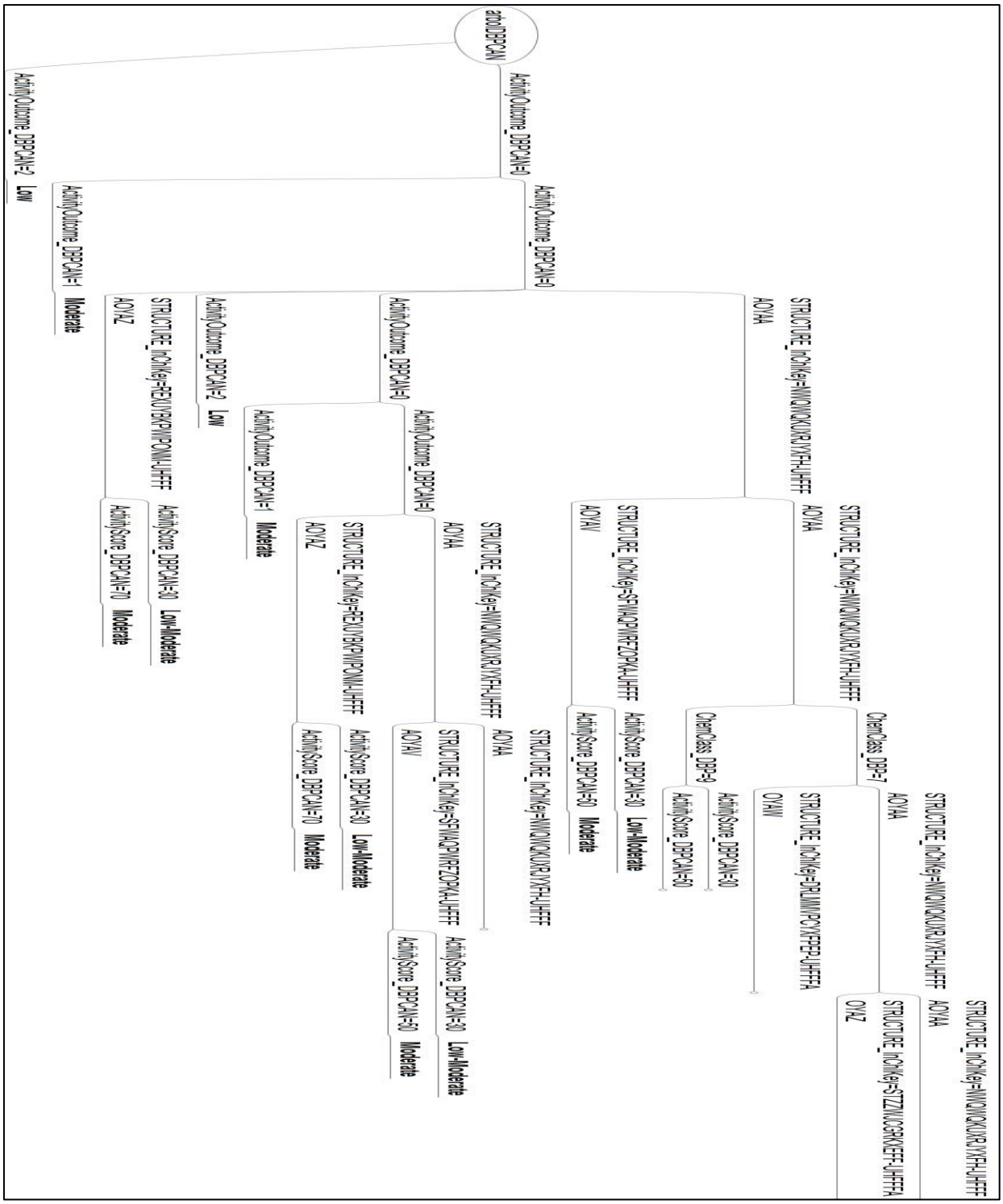
```
random = new Random(seed);
// Se baraja  el dataset

        dataSet1.randomize(random);

        double total = 0.0;
        for (int j_fold = 0; j_fold < folds; j_fold++) {
          //  Se realiza la validación cruzada...
          train = dataSet1.trainCV(folds, j_fold);
          test = dataSet1.testCV(folds, j_fold);
          if (trace) {
            System.out.println("Data for training ...");
            View.viewData(train);
          }
          /* aprendizaje del KNN*/
          Knn.setDistanceWeighting(new SelectedTag(Knn.WEIGHT_INVERSE,
                                                   Knn.TAGS_WEIGHTING));
        /* Se construye el clasificador*/
        Knn.buildClassifier(train);

          if (trace) {
        System.out.println(tree.toString());
        };
          System.out.println("clasificador de KNN    " + Knn.toString());

          if (Acc) {
            if (trace) {
              System.out.println("Data for testing ...");
              View.viewData(test);
            }
            enume = test.enumerateInstances();right = 0.0;
            while (enume.hasMoreElements()) {
              instance = (Instance) enume.nextElement();
              classified_class = Knn.classifyInstance(instance);
              real_class = instance.classValue();
              if (classified_class == real_class) {
                right++;
                if (trace)System.out.println(instance + " right");
              }
              else if (trace)
                System.out.println(instance + " mistake");
            }
            Accuracy[i_rep][j_fold] = (right / test.numInstances());
            Rights[i_rep][j_fold] = (int) right;

          }
          // Evaluacion del AUC
          if (AUC) cAUC[i_rep][j_fold] = computeAUC(Knn, test);
          System.out.println("AUC" + cAUC[i_rep][j_fold] + "\n");

        } ;
      };
```

C3.Función para tomar el valor dela distancia de edición de la matriz

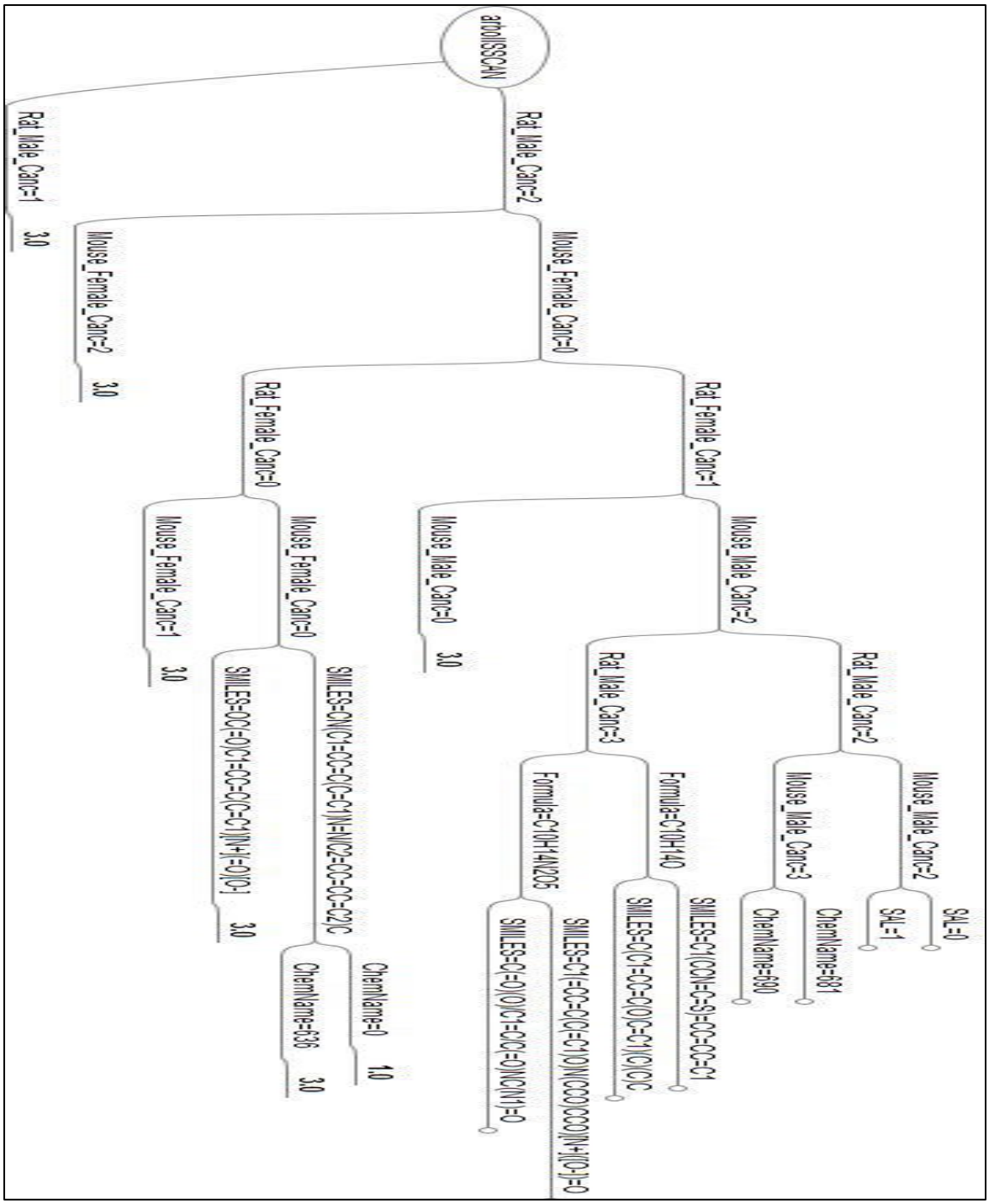
```
        protected double differenceFromString(int  index,double first,
double second)
 {
       // Se carga la matriz de distancia  con los valores calculados de
       la distancia de edición
        Instances DS=m_Train;
              double diff=-1;
               double x=-1, y=-1;

              String dsString;

                    int i=0;
                    //Se Verifica que las instancias sean diferentes
                    if (first!=second)
                    {
                          x=first;
                          y=second;

                    }
                    //Se toma de la matrix de distancia precalculada el
                    //valor de x,y para cada atributo
                    double[][] p=(double[][])
                    (ms.getMetricSpace()).elementAt(index);

                /// Se normalizan los valores de la matriz de distancia

               diff=0;
              if(x!=-1 && y!=-1)
              {


              diff= p[(int) x][(int) y];

              diff=nor(dif,index);
              /*Se realiza el equilibrio de la partición */
              if (diff < 0.5) {
                  diff = 1.0 - diff;
                }

              }

              return diff ;



 }

    }
```

Apéndice D. Arboles de Decisión generados por el DBDT

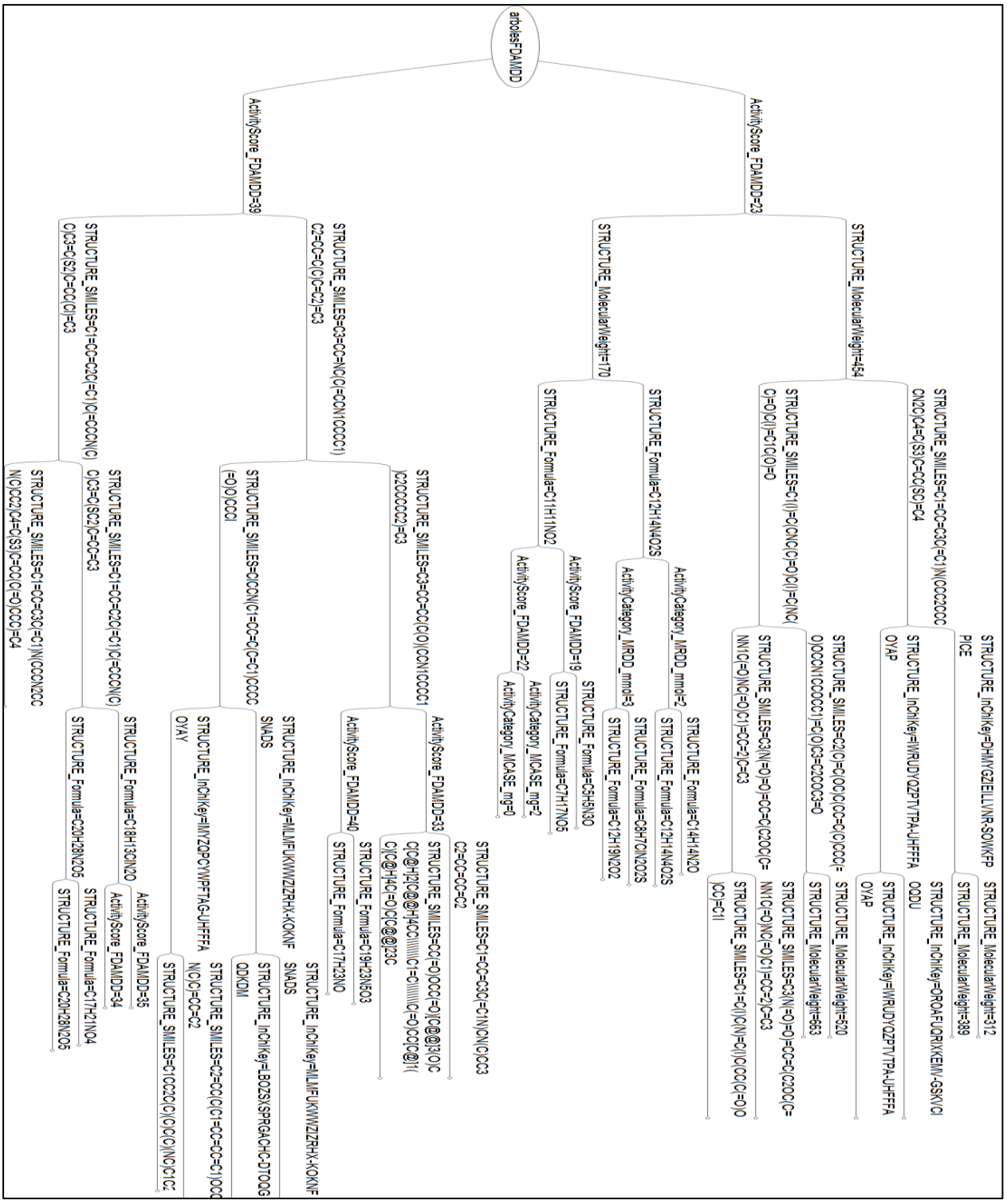
D1.Árbol de decisión DBPCANnostructures42



D2.Árbol de decisión ISSCANv2



D1.Árbol de decisión FDAMDD-3



Apéndice E. Ejemplos de Conjuntos de Datos.

E1.Ejemplo conjunto de datos DBPCAN

No.	STRUCTURE InChIKey	STRUCTURE Formula	STRUCTURE SMILES	Activity Outcome DBPCAN	STRUCTURE Molecular Weight	ActivityScore DBPCAN	ActivityConcern Level_ Carcinogenicity
1	NBBJYMSMWIIQGU-UHFFFAOYAZ	C3H6O	CCC=O	active	58.0	30.0	Low-Moderate
2	DTUQWGWMVIHBKE-UHFFFAOYAO	C8H8O	O=CCC1=CC=CC=C1	inconclusive	120.0	10.0	Moderate
3	XNGIFLGASWRNHJ-FLKJSBTCG	C8H6O4	OC(C1=CC=CC=C1C(O)=O)=O	inactive	166.0	0.0	Low
4	QMKYBPDZANOJGF-TUSFSZEUCZ	C9H6O6	O=C(O)C1=CC(C(O)=O)=CC(C(O)=O)=C1	inactive	210.0	0.0	Low
5	HPARLNRMYDSBNO-UHFFFAOYAC	C8H6O2	C1(C=CC=C2)=C2OC=CO1	active	134.0	30.0	Low-Moderate
6	WPYMKLBDIGXBTP-FZOZFQFYCI	C7H6O2	OC(=O)C1=CC=CC=C1	inactive	122.0	0.0	Low
7	JFDZBHWFFUWGJE-UHFFFAOYAY	C7H5N	N#CC1=CC=CC=C1	inconclusive	103.0	10.0	Moderate
8	SUSQOBVLVYHIEX-UHFFFAOYAJ	C8H7N	N#CCC1=CC=CC=C1	inactive	117.0	0.0	Low
9	RDQSIADLBQFVMY-UHFFFAOYAS	C14H20O2	CC(C)(C)C(C)(C)C(C)(C)C)=C1)=O)=CC1=O)C	inconclusive	220.0	10.0	Moderate
10	HIACAHMKXQESOV-UHFFFAOYAH	C12H14	C1=C(C(C)=C)C(C(C)=C)=CC=C1	inconclusive	158.0	10.0	Moderate
11	KDPAWGWELVVRCH-JLSKMEETCD	C2H3BrO2	O=C(O)CBr	inconclusive	139.0	10.0	Moderate
12	REXUYBKPWIPONM-UHFFFAOYAZ	C2H2BrN	BrCC#N	active	120.0	30.0	Low-Moderate
13	FNXLCIKXHOPCKH-UHFFFAOYAJ	H2BrN	NBr	inactive	96.0	0.0	Low
14	DRLMMVPCYXFPEP-UHFFFAOYAW	C7H4BrNS	BrC1=NC2=C(C=CC=C2)S1	active	214.0	30.0	Low-Moderate
15	UPSXAPQYNGXVBF-UHFFFAOYAN	C4H9Br	CC(Br)CC	active	137.0	30.0	Low-Moderate

E2.Ejemplo conjunto de datos FDAMDD

No.	STRUCTURE Formula	STRUCTURE Molecular Weight	STRUCTURE SMILES	STRUCTURE_InChIKey	ActivityScore FDAMDD	ActivityCat. MRDD_mmol	ActivityCat. MCASE_mg	Therapeutic Category
1	C13H18O2	206.0	CC(C1=CC=C(C=C1)CC(C)C)C(=O)O	HEFNNWSXXWATRW-YHMJCDSICY	19.0	Low	Low	art1
2	C16H14O3	254.0	C(C(=O)CCC(O)=O)C=C1)=C2 O=C(O)CC1=C(C(C=C3)= CC=C3Cl)N=C(C2=CC=CC=C2)S1	ZPAKPRAICRBAOD-GPQMBLKYCN	25.0	Low-Moderate	Low	art1
3	C17H12ClNO2S	330.0	C2=CC=CC(NC1=CC=CC(C	UYBDAZJACR-HCKMINDGCV	26.0	Low-Moderate	Low	art1
4	C15H15NO2	241.0	)=C1C)=C2C(=O)O	HYYBABOKPJLUIN-ACEWLPOYLGNNHV	23.0	Low	Low	art1
5	C19H23NO2	297.0	C2(CC(C)C)=CC=C(C(C)C(=O)O CC1=NC=CC=C1)C=C2	-UHFFFAOYAA	21.0	Low	Low	art1
6	C25H34O6	431.0	O=[C@]1[CO][C@]1([C@H]2C1) (O[C@@]1(CCC)[H	VOVIALXJUBGFJZ-VXKMTNQYBX	67.0	High	High	art1
7	C14H11Cl2NO 2	296.0	C2(CC(O)=O)=CC=CC=C2N C1=C(Cl)C=CC=C1Cl	DCOPUUMXTXDBNB-GPQMBLKYCR	32.0	Moderate	Moderate	art1
8	C16H19ClO3	295.0	C2CCCC(C1=C(Cl)C=C(C(=O) CCC(O)=O)C=C1)C2	IJTPQQVCKPZIMV-LILDFLRNCN	24.0	Low	Low	art1
9	C20H20N2O2	320.0	C3[=O]C(CC=C(C)C)C(=O)N (C1=CC=CC=C1)N3C2=CC=CC=C2	RBBWCVQDXDFISW-UHFFFAOYAV	27.0	Low-Moderate	Low	art1
10	C19H20N2O3	324.0	N3C(C1=CC=CC=C1)=C (C2=CC=CC=C2)OC=3N(CCO)CCO	UUCMDZWCRNZCOY-UHFFFAOYAE	25.0	Low-Moderate	Low	art1
11	C13H14ClNO2	252.0	C2(Cl)=C(N1CCC=CC1)C=CC (C(C)C(O)=O)=C2	PIDSZXPFGCURGN-WYUMXYHSCP	25.0	Low-Moderate	Low	art1
12	C20H17FO3S	356.0	C3=C(S(C)=O)C=CC(C=C1C(C)= C(CC(O)=O)C2=C1C=CC(F)=C2)=C3	MLKXDPUZXIRXEP-VHKGTHMQDT	30.0	Moderate	Low	art1
13	C18H18N2O	278.0	C1(=CC=C2C(=C1)N(C(\\\\\\\N=C/2C3=CC=CC=C3)=O)C(C)C	JTIGKVIOEQASGT-UHFFFAOYAZ	24.0	Low	Low	art1
14	C14H12O3S	260.0	C2=CC=CC(=O)C1S C(C(C)C(=O)O)=CC=1)=C2	GUHPRPJDBZHYCJ-WYUMXYHSCF	27.0	Low-Moderate	Low	art1
15	C19H20N2O2	308.0	O=C1N(C2=CC=CC=C2) N(C3=CC=CC=C3)C(=O)C1CCCC	VYMDGNCVAMGZFE-UHFFFAOYAL	28.0	Low-Moderate	Low	art1
16	C21H18ClNO6	416.0	C1=C(OC)C=C3C(=C1) N(C(=O)C2=CC=C(Cl)C=C2)C(C)=C3CC(=O)	FSQKKOOTNAMONP-LQFNOIFHCH	35.0	Moderate	Moderate	art1
17	C15H13NO2S	271.0	C1=CC=C2C(=C1)N(C)C3=C(S2)C=CC(CC(O)=O)=C3	LMINNBXUMGNKMM-HCKMINDGCO	26.0	Low-Moderate	Low	art1

E3.Ejemplo conjunto de datos ISSCAN

No.	Mol-Weight	Formula	SMILES	SAL	Rat Male Canc	Rat Female Canc	Mouse Male Canc	Mouse _Female Canc	Rat_ Male NTP	Rat Female NTP	Mouse Male_ NTP	Mouse Female NTP	Canc
1	225.0	C14H15N3	CN(C1=CC=C(C=C1)N=N/C2=CC=CC=C2)C	3.0	ND	3.0	3.0	1.0	ND	ND	ND	ND	1.0
2	223.0	C15H13NO	O=C(Nc3c2c1ccccc1C2ccc3)C	3.0	ND	1.0	ND	ND	ND	ND	ND	ND	3.0
3	202.0	C16H10	c(c(c(cc1)ccc2)c2cc3)(c1ccc4)c34	3.0	ND	ND	1.0	1.0	ND	ND	ND	ND	3.0
4	119.0	CHCl3	ClC(Cl)Cl	1.0	3.0	3.0	3.0	ND	ND	ND	ND	ND	3.0
5	268.0	C18H20O2	Oc2ccc(/C(=C(/c1ccc(O)cc1)CC)CC)cc2	1.0	3.0	3.0	3.0	ND	ND	ND	ND	ND	3.0
6	190.0	C9H6N2O3	N(=O)(=O)c1ccn(=O)c2ccccc12	3.0	3.0	3.0	3.0	ND	ND	ND	ND	ND	3.0
7	143.0	C10H9N	C12=C(C=CC(=C1)N)C=CC=C2	3.0	ND	3.0	3.0	ND	ND	ND	ND	ND	3.0
8	184.0	C12H12N2	NC1=CC=C(C2=CC=C(N)C=C2)C=C1	3.0	3.0	3.0	3.0	ND	ND	ND	ND	ND	3.0
9	223.0	C15H13NO	O=C(Nc3ccc2c1ccccc1Cc2c3)C	3.0	3.0	3.0	3.0	ND	ND	ND	ND	ND	3.0
10	252.0	C20H12	C1=CC2=CC=CC3=CC=C4C(=C23)C1	3.0	3.0	3.0	1.0	ND	ND	ND	ND	ND	3.0
11	130.0	H6N2O4S	O=S(=O)(O)O.NN	3.0	3.0	3.0	3.0	ND	ND	ND	ND	ND	3.0
12	179.0	C6H18N3OP	O=P(N(C)C)(N(C)C)N(C)C	1.0	3.0	3.0	ND	ND	ND	ND	ND	ND	3.0
13	198.0	C12H10N2O	O=NN(C1=CC=CC=C1)C2=CC=CC=C2	1.0	3.0	3.0	1.0	1.0	ND	ND	ND	ND	3.0
14	102.0	C3H6N2S	S=C1NCCN1	1.0	3.0	3.0	3.0	CE	CE	CE	CE	CE	3.0
15	86.0	C4H6O2	CC1CC(=O)O1	3.0	3.0	3.0	3.0	ND	ND	ND	ND	ND	3.0
16	162.0	C10H10O2	C=CCC1=CC=C2C(=C1)OCO2	1.0	3.0	3.0	3.0	ND	ND	ND	ND	ND	3.0