Documentación del algoritmo para la construcción del árbol de decisión del cálculo de Triage - TAppitree	
Elaborado por	Luisa Álvarez Valencia
Objetivo	Este documento tiene como objetivo explicar el funcionamiento y la lógica detrás del árbol de decisión de los síntomas de la aplicación. Cabe resaltar que esto es una librería y es aparte de WSTappi.
Información general	IP: 200.58.126.15 Puerto: 4848, accede a Glassfish en modo administrador. Puerto: 8080, accede a los servicios del servidor.
Contenido	

1. Explicación teórica

Para el desarrollo del Triage tentativo en la aplicación es necesario el uso de un árbol de decisión el cual determina si un síntoma o no está presente en la dolencia de un paciente. El aprendizaje de árboles de decisión es uno de los métodos más utilizados para inferencia inductiva, en nuestro caso para determinar el valor de Triage dado unos determinados síntomas. Los arboles utilizan algoritmos de aprendizaje como ID3 (interactive dichotomizer 3) [1], [2], CART y C4.5. La representación de los árboles de decisión es una manera de clasificar instancias de manera que se ordenen desde la raíz hasta la hoja, donde la hoja provee la clasificación de la instancia.

El aprendizaje de árboles de decisión es más que todo utilizado cuando los problemas contienen las siguientes características: las instancias tiene un valor fijo, cuando se mueve la información por el árbol esta tiene valores boléanos (de si o no), las instancias representan

descripciones disyuntivas, los datos de entrenamiento pueden contener errores o no estar completos [1] en el caso de ciertos algoritmos ya que ID3 necesita la información completa.

A continuación, se describirán los diferentes algoritmos con mayor detalle. ID3 es no incremental, lo que significa que este deriva de una serie de clases fijas y unos datos de entrenamiento determinados, este algoritmo es inductivo, lo que significa que dado una instancia determinada de las clases el árbol que se crea debe funcionar para todas las futuras creaciones [3]. Por otro lado, C4.5 es una extensión de ID3[4], la diferencia más grande es que ID3 utiliza separaciones binarias mientras que C4.5 [5]presente multiseparaciones, ambos utilizan entropía como medida de inducción (ID3 se desarrolló en 1986 y C4.5 en 1993) [6]. C4.5 mejora ciertos aspectos de ID3 como que permite tener datos continuos, datos faltantes en el árbol, permite que los diferentes atributos tengan pesos, quita replicas una vez el árbol es armado.

La entropía se define como la cantidad de información proveída por un evento: a mayor probabilidad del evento la entropía es baja (rara), a menor probabilidad más información provee y la entropía es alta [7]. La ecuación de la entropía significa que es la suma de probabilidades de que una instancia suceda por la probabilidad logarítmica en base dos de que esa instancia suceda.

$$Entropie(P) = -\sum_{i=1}^{n} p_i \times log(p_i)$$

[7, p. 5]

También se debe tener en cuenta la ganancia, esto es el grado de mezclar todas las clases para todas las muestras en cualquier posición del árbol.

$$Gain(p,T) = Entropie(p) - \sum_{j=1}^{n} (p_j \times Entropie(p_j))$$

Donde Pi es uno de los valores posibles que pude tomar el tributo T [7, p. 5].

CART (Classification and Regression Tree), es un método flexible que describe como una variable Y después de asignar en un vector X unas posibilidades para los nodos. Este modelo utiliza un árbol binario para dividir las posibilidades en las cuales Y sean continuamente parejos. Este tipo utiliza el índice GINI para saber que rama como atributo debe ser generada, se debe elegir la cual tenga el índice GINI más pequeño, además CART utiliza exámenes que aproximan el resultado cuando los atributos no tienen valor conocido[7, p. 5].

Los arboles utilizan datos de entrenamiento (son los datos utilizados para armar el modelo [8]) y datos de prueba (se utiliza cuando se finaliza el ensamblaje del modelo para evaluar que tan bien conformado estuvo este [8]). Los cuales funcionan como instancias de la información que se va a colocar en los nodos.

2. Diagrama de clases

En esta sección se observa el diagrama de clases que se utilizará para el desarrollo del árbol del Triage.

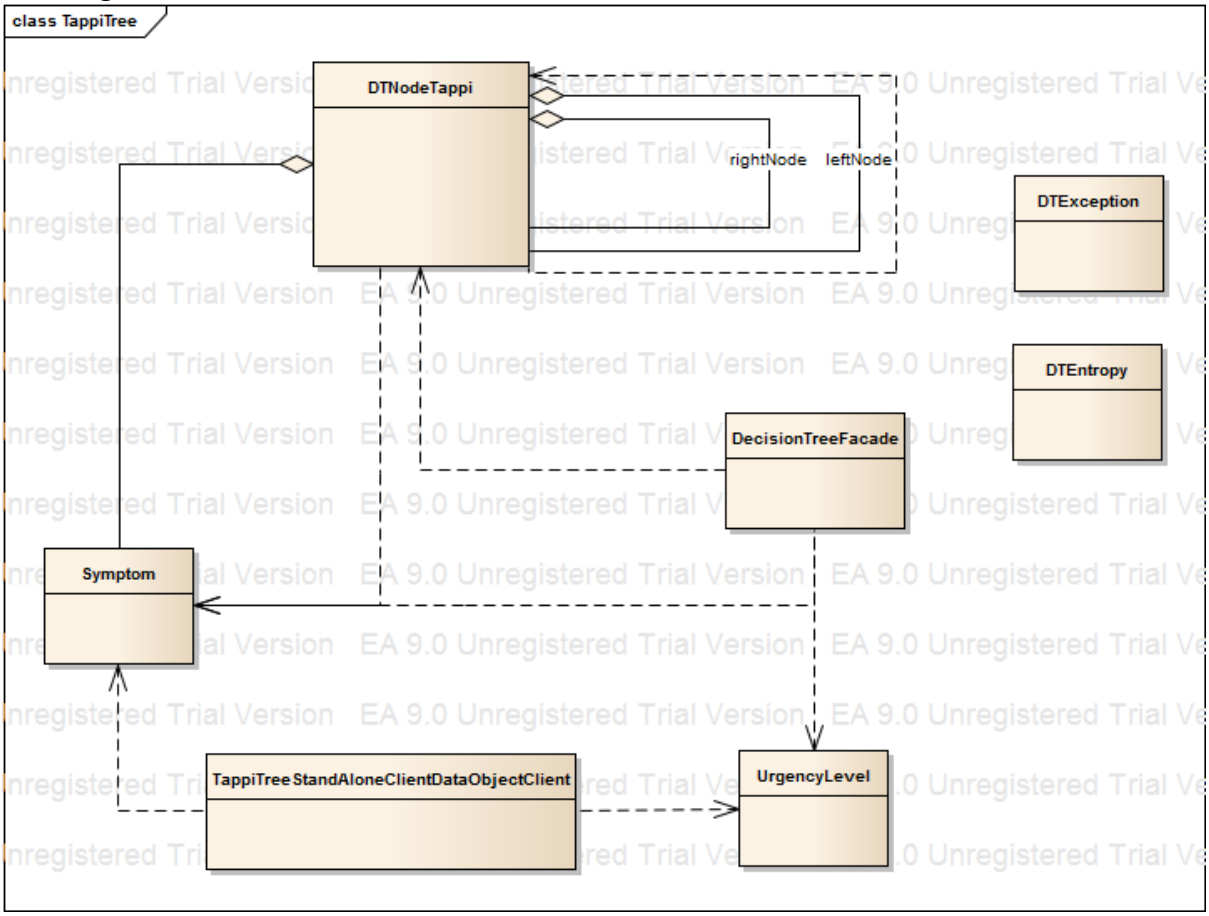


Ilustración 1 diagrama clases tappi tree

A continuación se encuentran unas tablas explicativas, mostrando cada una de las diferentes clases utilizadas en el árbol para la generación del árbol y sus interacciones.

ID	**1**	**Elemento del Dominio**	**Symptom**
Descripción	Representación de los síntomas en el árbol		
Atributos			
Nombre		Descripción	Tipo de Dato
Idsymptom		Indica el número de identificación del síntoma.	Entero
sympName		Indica el nombre del sintoma	Cadena de caracteres
symptomValues		Indica la presencia o ausencia del síntoma.	Arreglo de cadena de caracteres, en la primera

		posición almacena si y en la segunda no.
Objetivo	Identificar los síntomas con los cuales se relaciona el árbol.	

ID	**2**	**Elemento del Dominio**	**UrgencyLevel**
Descripción	Representación el nivel de urgencia asociado al valor del síntoma.		
Atributos			
Nombre		Descripción	Tipo de Dato
triageindicator		Indica el valor del Triage el cual puede variar entre 1 y 5.	Cadena de caracteres
userSymptom		Indica la presencia o ausencia del síntoma, de un usuario	Arreglo de cadena de caracteres, en la primera posición almacena si se tiene el primer síntoma y así sucesivamente.
Objetivo	Identificar el nivel de urgencia a la cual pertenecen los síntomas con los cuales se relaciona el árbol.		

ID	**3**	**Elemento del Dominio**	**TappiTreeStandAloneClientDataObjectClient**
Descripción	Representación del objeto que crea el cliente para interactuar con el árbol		
Atributos			
Nombre		Descripción	Tipo de Dato
log		Se utiliza para ver lo que sucede en la clase.	Log
Objetivo	El objetivo de esta clase es coger los datos de nivel urgencia y síntomas y comunicárselos a la fachada del árbol		
Relaciones			
Symptom	Esta clase depende de síntomas para poder crear una lista de estos.		
NivelUrgencia	Esta clase depende de nivel de urgencia para poder tener elementos de entrenamiento y es por ello que se debe relacionar con esta.		

ID	**4**	**Elemento del Dominio**	**DecisiónTreeFacade**
Descripción	Representación la fachada del árbol de decisión		
Atributos			
Nombre		Descripción	Tipo de Dato

log	Se utiliza para ver lo que sucede en la clase.	Log
Objetivo	El objetivo de esta clase es coger los datos de ObjectClient y los arma a manera de árbol	
Relaciones		
nivelUrgencia	Requiere de nivel urgencia para construir el árbol.	
Symptom	Requiere de síntomas para construir el árbol e ingresar estos en los nodos.	
DTNodeTappi	Requiere de los nodos para armar el árbol con las interacciones y conexiones entre nodos como se debe.	

ID	**5**	**Elemento del Dominio**	**DTNodeTappi**
Descripción	Representación de los nodos y la construcción del árbol.		
Atributos			
Nombre		Descripción	Tipo de Dato
leftNode		Indica el nodo de la izquierda del árbol del nodo padre.	DTNodeTappi
rightNode		Indica el nodo de la derecha del árbol del nodo padre.	Cadena de caracteres
sympton		Indica un síntoma en dado caso de que no sea una hoja el nodo.	Symptom
log		Permite ver lo que sucede en el árbol a lo largo de la ejecución.	log
outputLabel		Indica el nivel de Triage en dado caso de que sea una hoja, es un número de 1 a 5.	Cadena de caracteres
nodeType		Indica si el nodo es un nodo de síntomas o una hoja.	Tipo de nodo
Objetivo	Identifica cada uno de los nodos del árbol		

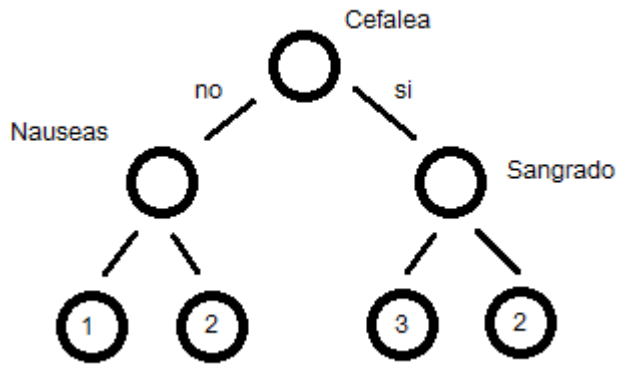
Relaciones	
DTNodeTappi	La clase depende de sí misma para crear un árbol.
rightNode	La clase tiene una interacción con si misma a modo de lista para tener los nodos de la derecha.
leftNode	La clase tiene una interacción con si misma a modo de lista para tener los nodos de la izquierda.
Symptom	La clase tiene una lista de síntomas.
Symptom	La clase depende de síntomas para la creación de los nodos.

ID	6	Elemento del Dominio	DTEntropy
Descripción	Calcula la entropía de las diferentes instancias.		
Objetivo	Armar el árbol de acuerdo a los datos suministrados.		

ID	7	Elemento del Dominio	DTException
Descripción	Es una clase que maneja las excepciones que se generen del árbol		
Atributos			
Nombre		Descripción	Tipo de Dato
problema		Indica la descripción de alguna excepción generada.	Cadena de caracteres
Objetivo	Almacena y muestra las excepciones		

3. Construcción

Se evalúa el estado del arte del cual se extrae la información para la creación de las instancias de los síntomas. En la parte inferior se explicará con un ejemplo la construcción y uso del árbol.



La tabla de NivelUrgencia tiene los siguientes datos

- 1 No no
- 2 no si
- 3 si no
- 2 si si

Por lo tanto estos datos son los datos de entrenamiento para armar el árbol.
La tabla de síntomas tiene lo siguiente:

- 1 Cefalea
- 2 Nauseas
- 3 Sangrado

TappiTreeStandAloneClientDataObjectClient este tiene como finalidad generar una relación entre nivel de urgencia y síntomas.

DecisiónTreeFacade es la fachada que tiene las órdenes para generar el árbol gracias a la interacción de los objetos.

3.1. Recorrido

Cuando un paciente llega con una dolencia este por ejemplo menciona que sólo tiene nauseas por lo tanto su ingreso de síntomas debe ser no (cefalea), si (nauseas) por lo tanto su Triage tentativo es 2. Este resultado del triage y recorrido lo lleva a cabo WSTappi.

4. Diagrama de secuencia, comportamental

El siguiente diagrama de secuencia muestra las interacciones entre las clases y como se lleva a cabo el cálculo tentativo del Triage.

5. Integración con la lógica de negocio

WSTappi se conecta con Tappitree (.jar) como una librería que contiene ya la conformación final del árbol completo realizado con la lógica de negocio y la persistencia de las tablas (datos de entrenamiento). Al tener esta unión con el WAR de la aplicación se obtiene el resultado del Triage tentativo a partir de los síntomas que el paciente ingrese, de manera automática se recorren los nodos del árbol hasta llegar a las hojas. WSTappi recibe el árbol en un archivo XML el cual carga en memoria y recorre con ayuda de Tappitree para brindar el nivel de Triage.

6. Archivos fuentes

El archivo fuente se encuentra en:

7. Bibliografía

[1] T. M. Mitchell, *Machine learning*, International ed., [Reprint.]. New York, NY: McGraw-Hill, 20.

[2] M. Umanol, H. Okamoto, I. Hatono, H. Tamura, F. Kawachi, S. Umedzu, y J. Kinoshita, «Fuzzy decision trees by fuzzy ID3 algorithm and its application to diagnosis systems», en

Fuzzy Systems, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the Third IEEE Conference on, 1994, pp. 2113–2118.

[3] M. S. S.Vijayarani, «An Efficient Technique for Privacy Preserving Decision Tree Learning», vol. 3, n.º 9, p. 4, sep. 2013.

[4] M. M. Mazid, A. S. Ali, y K. S. Tickle, «Improved C4. 5 algorithm for rule based classification», en *Proceedings of the 9th WSEAS international conference on Artificial intelligence, knowledge engineering and data bases*, 2010, pp. 296–301.

[5] L. Tanner, M. Schreiber, J. G. Low, A. Ong, T. Tolfvenstam, Y. L. Lai, L. C. Ng, Y. S. Leo, L. T. Puong, S. G. Vasudevan, y others, «Decision tree algorithms predict the diagnosis and outcome of dengue fever in the early phase of illness», *PLoS Negl Trop Dis*, vol. 2, n.º 3, p. e196, 2008.

[6] Songul Cinaroglu, «Comparison of Performance of Decision Tree Algorithms and Random Forest: An Application on OECD Countries Health Expenditures», vol. 138, n.º 1, mar. 2016.

[7] B. HSSINA, A. Merbouha, H. Ezzikouri, y M. Erritali, «A comparative study of decision tree ID3 and C4.5», *Int J Adv Comput Sci Appl*, vol. 4, n.º 2, 2014.

[8] University of Notre Dame, «Data Mining Definitions». [En línea]. Disponible en: https://www3.nd.edu/~busiforc/handouts/DataMining/dataminingdefinitions.html. [Accedido: 14-sep-2016].