PROP: REGLES D'ASSOCIACIÓ

1r entrega

MARTIN GALINDO, PAU SEBASTIA

MIKAUTADZE, GIORGI

PLA ORTIZ DE ZARATE, ANDREU

SOLANA FRANCH, ÒSCAR

pau.sebastia.martin@est.fib.upc.edu

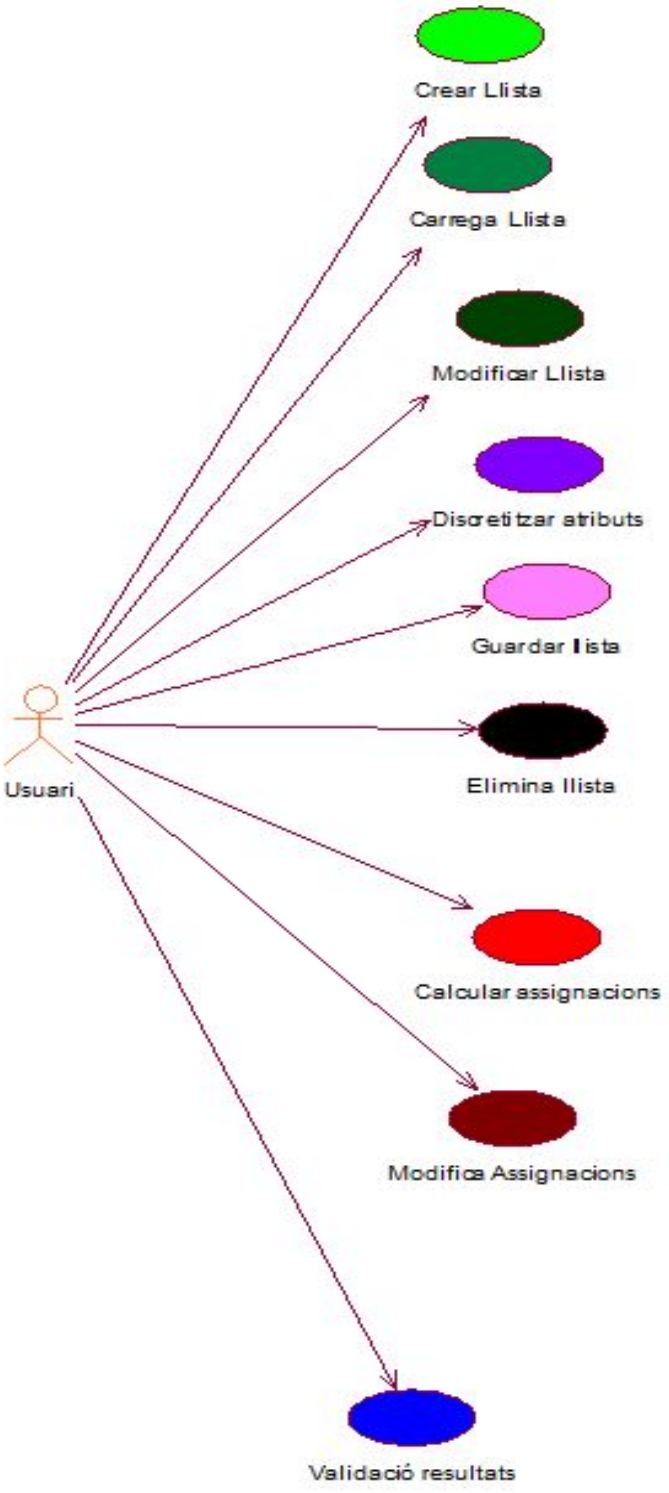
giorgi.mikautadze@est.fib.upc.edu

andreu.pla.ortiz.de.zarate@est.fib.upc.edu

oscar.solana@est.fib.upc.edu

INDEX

Esquema casos d'ús (Funcionalitats principals)



CASOS D'ÚS: Explicació funcionalitats

Crear llista:

Aquesta funcionalitat crea una llista buida dels quals s'hauràn d'inicialitzar els atributs i els casos

CREA LLISTA:

Pre: La llista no existeix

Post: Aquesta funcionalitat crea una llista buida

INICIALITZA LLISTA

Pre: Els valors per inicialitzar la llista estàn en el format que es demana

Post: Aquesta funcionalitat inicialitza la llista als valors que l'usuari li assigni a la llista

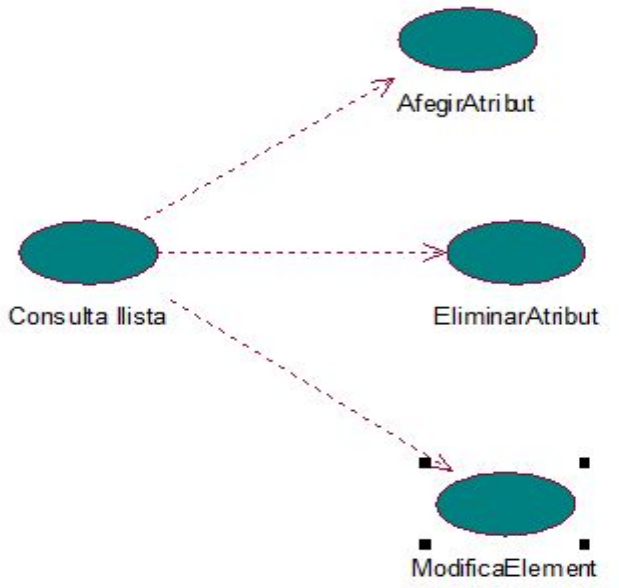
Carrega llista:

CARREGA LLISTA:

Pre: Existeix Fitxer amb path correcte

Post: Aquesta funcionalitat carrega una llista que està emmagatzemada en un fitxer.

Modificar llista:
Aquesta funcionalitat genera una sèrie d'accions per modificar la llista passada per paràmetre



CONSULTA LLISTA (//encara no implementat)

Pre: Existeix la llista

Post:Facilitar la visualització de les dades amb possibilitat de canvi.

AFEGIR ATRIBUT

Pre: -

Post: Afegeix un atribut a la taula i cada cas té el mateix atribut per aquell valor (passant per paràmetre el nou atribut i el valor desitjat).

ELIMINAR ATRIBUT

Pre: Existeix atribut

Post: Elimina un atribut de la taula i els seus valors per cada cas.

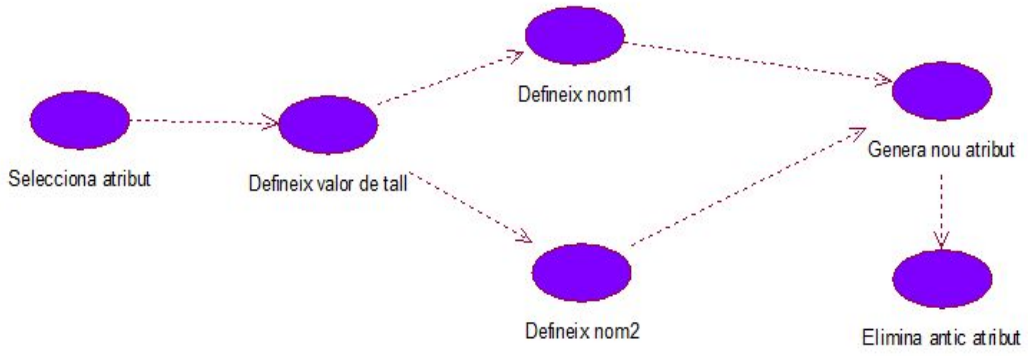
MODIFICAR ELEMENT

Pre: l'atribut donat es troba a la llista d'atributs

Post: Modifica el valor d'un dels paràmetres de l'atribut.

Discretitzar atributs:

Aquesta funcionalitat discretitza l'atribut en dos o més atributs de booleà



SELECCIONA ATRIBUT

Pre: L' atribut a voler seleccionar existeix

Post: Es selecciona l'atribut que l'usuari demana.

DEFINEIX VALOR TALL

Pre: El valor a donar ha de ser vàlid

Post: Es defineix un valor de tall per a l'atribut seleccionat

DEFINEIX NOM1

Pre: -

Post: Es defineix el nom de l'atribut per sota del valor de tall

DEFINEIX NOM2

Pre: -

Post: Es defineix el nom de l’atribut per sobre del valor de tall

GENERA NOU ATRIBUT

Pre: -

Post: Es discretitza l'atribut en dos o més atributs de booleà.

ELIMINA ANTIC ATRIBUT

Pre: -

Post: S’elimina l'atribut inicialment seleccionat

Guardar llista:

GUARDAR LLISTA:

Pre: -

Post: Aquesta funcionalitat confirma l'acció o les accions que l'usuari ha fet prèviament.

Elimina llista:

ELIMINA LLISTA:

Pre: La llista a eliminar existeix

Post: Aquesta funcionalitat elimina la llista que l’usuari indica.

Calcula assignacions:

Es calculen les associacions d'una llista determinada. Tenen lloc les eleccions dels paràmetres per tal de fer el calcul de les regles associades a l'algorisme Apriori. És recopilen les diverses assignacions amb les frequències i confiances obtingudes.

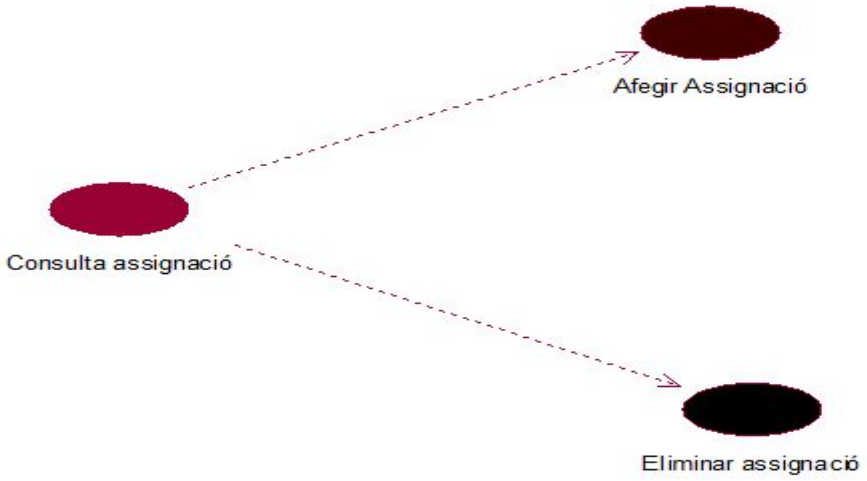
DEFINICIÓ PARÀMETRES:

Pre: La definició de paràmetres han de ser valors entre 0 i 1.

Post: L'usuari proporciona la frequència i confiança per trobar les regles associades

Modifica assignacions:

Aquesta funcionalitat edita les assignacions donades per l'usuari



CONSULTA ASSIGNACIÓ:

Pre: Ha d'existir l'assignació que l'usuari vol consultar

Post: Aquesta funcionalitat visualitza el contigut de l'assignació que l'usuari sol·licita

AFEGIR ASSIGNACIÓ:

Pre: La nova assignació amb nous paràmetres han de ser valors entre 0 i 1

Post: Aquesta funcionalitat afegeix una nova assignació (nova frequencia i nova confiança).

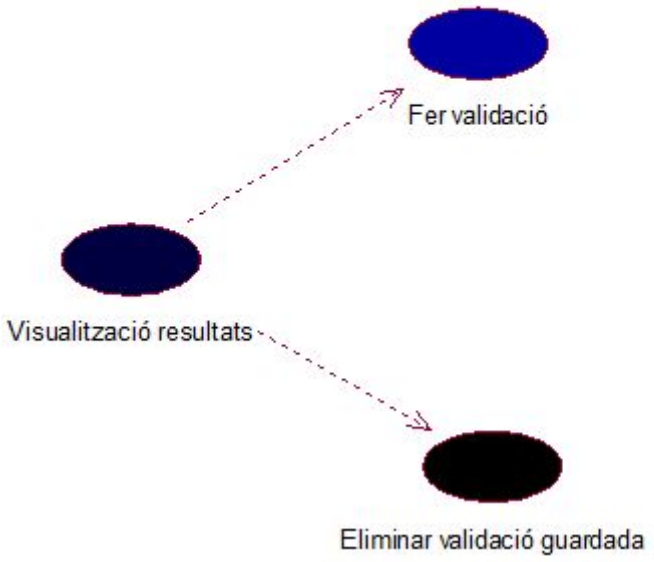
ELIMINAR ASSIGNACIÓ:

Pre: Existeixen assignació i llista

Post: Aquesta funcionalitat elimina l'assignació marcada per l'usuari

Validació resultats:

Aquesta funcionalitat ens permet visualitzar i validar el conjunt de regles associades de les llistes i les seves respectives assignacions.



VISUALITZACIÓ RESULTATS:

Pre: Existeixen assignació i llista

Post: Aquesta funcionalitat ens permet visualitzar les regles d'associacions.

FER VALIDACIÓ:

Pre: Existeixen assignació i llista

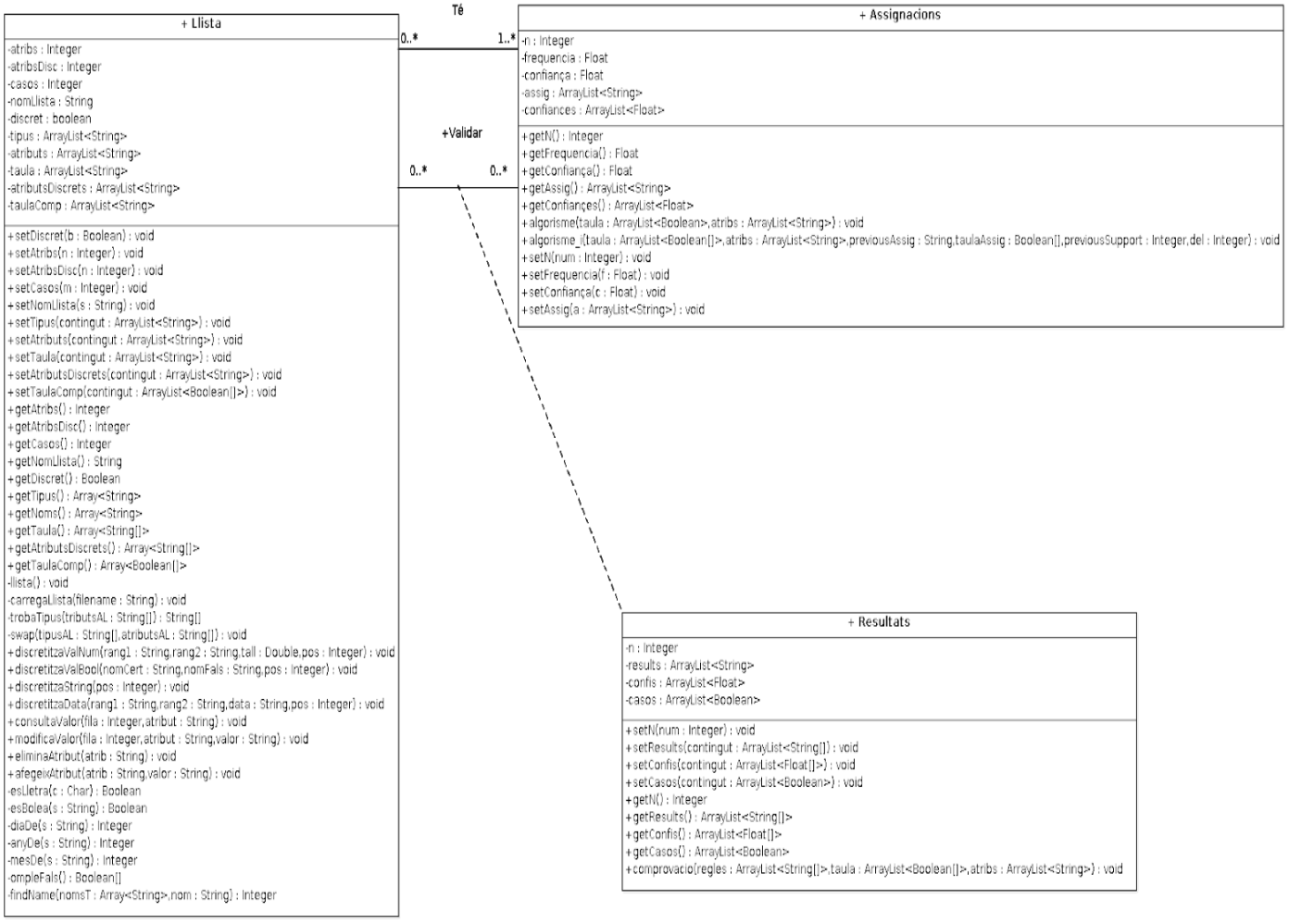
Post: Aquesta funcionalitat agafa les assignacions i l'aplica al exemple donat

ELIMINAR VALIDACIÓ GUARDADA:

Pre: Existeixen assignació i llista

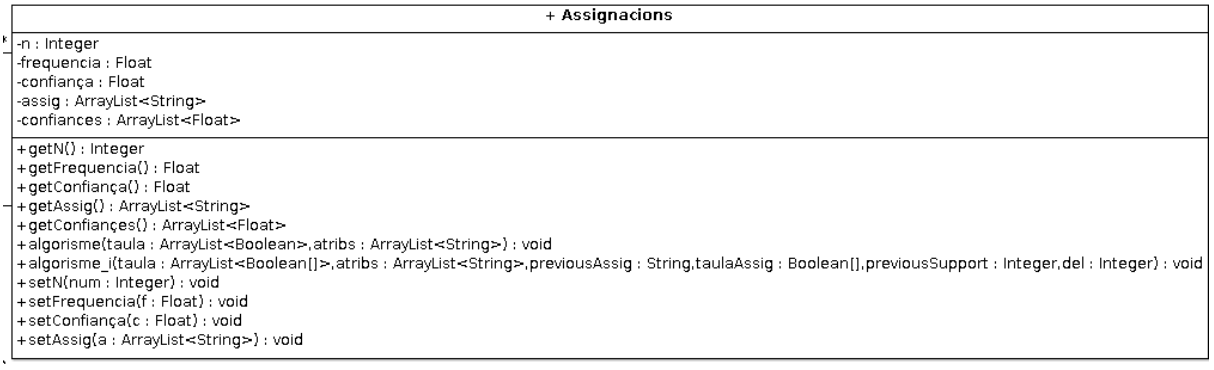
Post: Aquesta funcionalitat elimina les regles d'associacions donades.

DIAGRAMA UML: ESQUEMA



LLISTA

ASSIGNACIONS



RESULTATS

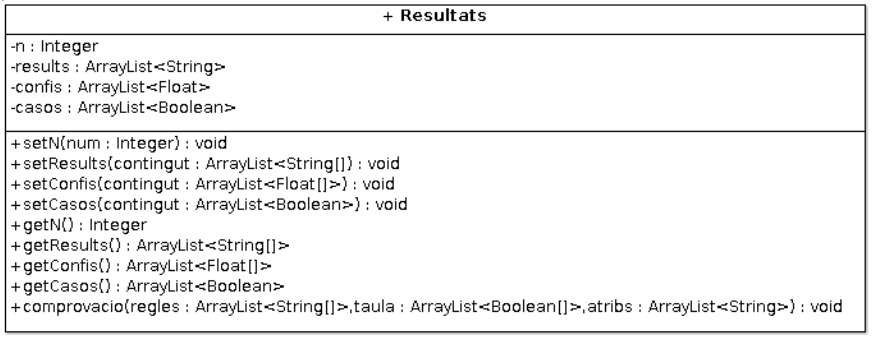


DIAGRAMA UML: EXPLICACIÓ DE CADA CLASSE

Hem decidit organitzar el projecte en 3 classes:

- **Llista**
- **Assignacions**
- **Resultats**

LLISTA

La classe Llista es la primera encarregada del tractament de les dades. Rep el fitxer o els diferents registres dels atributs de les dades d'entrada ii fa una primera manipulació per poder simplificar la fiabilitat i tractament d'aquestes dades.

El tractament de les dades es descompon en organitzar les dades ii transformar-les d'una manera homogènia per simplificar el seu posterior càlcul.

En aquesta classe es produirà la correctesa dels atributs d'entrada, permetent una primera manipulació de canvis i de discretització per fer la entrada d'atributs definitiva

Cada Llista podrà tenir varies classes Assignacions amb les que podrem consultar la classe Resultats

ASSIGNACIONS

En aquesta classe hem decidit que es pugui tractar les entrades en l'elecció dels paràmetres de la freqüència i la confiança per part de l'usuari; així pot tenir sobre una mateixa entrada d'atributs, diferents càlculs amb aquests paràmetres i poder guardar-los i consultar-los d'una forma més ordenada.

Per tant, al tenir els paràmetres de crida a escollir, serà la classe que implementarà l'algorisme principal Apriori.

Farà el còmput dels fitxer d'entrada de la classe Llista amb els paràmetres de la classe Assignacions.

RESULTATS

Aquesta classe té la funció principal de consultar i validar els resultats obtinguts amb les regles obtingudes de la relació de les anteriors classes. La opció de validació s'efectuarà comprovant si és pot adaptar a un altre model d'entrada amb tipus d'atributs similars.

DESCRIPCIÓ DE LES ESTRUCTURES DE DADES

(Classe LLISTA)

FUNCIONALITATS AUTOMÀTIQUES DE PRE-PROCÉS DE LES DADES.

En aquestes funcionalitats no intervé l'usuari. Utilitzem unes estructures per poder manipular correctament les dades d'entrada de l'usuari.

Hem escollit utilitzar la entrada del fitxer de dades en format "strings" que utilitzarem primerament per obtenir les dades que ens proporcionarà l'usuari. Sobre aquests strings farem les pertinents classificacions de les entrades dels atributs en els diferents tipus de dades. Les dades les diferenciarem més tard entre strings, booleans, doubles ii enters.

La nostra proposta, finalment, ha estat fer majoritàriament:

ArrayList, *que pot ser del tipus <String>, <String[]> , <Boolean>.*

Amb aquesta eina diferenciarem entre la entrada del tipus de dades de dades ii el seu valor, que organitzarem en una Matriu de strings, fent de cada fila els registres a tractar.

La idea és transformar aquesta Matriu de strings, de les primeres dades, en una Matriu de Booleans, amb la qual tindrem més facilitat per la posterior manipulació del suport ii la confiança.

Per obtenir aquesta Matriu de booleans, abans hem d'obtenir una matriu que té cada fila amb un ArrayList de cada entrada del fitxer per ordre.

Per exemple: Si tenim de registre "string1, int1, string2, booleans, int2", llavors tindrem les files organitzades en

fila0 amb tots string1

fila1 amb tots in1

fila2 amb tots string2

etc..

independentment si tenen un valor null o no.

El motiu d'aquesta organització és facilitar el posterior càlcul dels candidats amb el suport ii confiança correctes.

(Classe ASSIGNACIONS)

INDUCCIÓ REGLES ASSOCIACIÓ

Les regles d'associació sol.licitades per aquest projecte les obtindrem a partir dels paràmetres n de freqüència i la confiança desitjades per l'usuari.

Encapsularem totes les assignacions satisfactòries en estructures de dades

ArrayList<String>

ArrayList de <Float>

ii utilitzarem la matriu de booleans amb les dades obtingudes ii transformades a partir del fitxer d'entrada.

Algorisme s'encarrega de buscar els candidats amb suport i confiança vàlides. Per fer-ho, primer ha d'identificar tots els registres on els seus atributs són candidats a ser escollits ii comptarà el nombre de vegades que aquests atributs son certs.

Seleccionarem aquells atributs que compleixen una freqüència major o igual a la desitjada ii per cada un dels atributs candidats, comprovem les combinacions que compleixen la freqüència desitjada.

Finalment es fa una cerca en arbre per recórrer només aquells casos amb una freqüència desitjable.

Aquest resultat s'assoleix mitjançant diverses crides recursives en l'algorisme, .

(Classe RESULTAT)

VALIDACIÓ DEL CONJUNT DE LA REGLA

Les estructures de dades utilitzades

ArrayList<String[]> ,

ArrayList<Boolean[]>

Per fer la validació agafarem les assignacions trobades i les aplicarem a exemples donats.

La finalitat es comparar la sortida de les regles d'associació d'una assignació amb entrada de una altre fitxer de registres, amb mateix nombre d'atributs, mateixa freqüència i confiança, però amb temàtica diferent. Per aconseguir-ho és necessària una nova matriu de booleans ii procedir de forma similar comparant amb strings, ja que el resultat de les assignacions estan guardats en strings. Comprovar cas per cas si es compleixen les assignacions ii quan es compleixen donar el resultat

DESCRIPCIÓ DE L'ALGORISME APRIORI

Per tal de trobar les regles d'associació donat un llistat d'atributs, hem decidit fer la següent implementació. Primer agafem els atributs discretitzats i n'identifiquem aquells que compleixen una freqüència superior o igual a la requerida, la combinació dels quals ens permetrà formar les regles.
Per a cada un dels atributs, apliquem la segona part de l'algorisme, que ens permet anar concatenant tots els conjunts d'atributs $\mathfrak{E}$-Freqüents obtenint així un llistat amb totes les regles possibles on l'últim element de cada regal és l'element implicat.

ALTERNATIVES D'OPTIMITZACIONS EN PROPERES ENTREGUES

(1) Un cop validades les regles, s'hauria de comprovar si tenen validesa en altres atributs. Això implica una tenir una altra matriu de casos amb els mateixos atributs amb la qual poder comprovar les assignacions trobades.

Per fer això podem utilitzar la propia matriu amb els atributs discretitzats per assegurar-nos de que no hi ha diferències entre aquests atributs.

Aquesta matriu, en el moment d'inserir les regles, podríem particionar-la en 2 parts, però tenint en compte els propòsits de tal separació:

- una part per de la matriu per inferir les regles (calcular associacions).

- l'altra part de la matriu per validar regles.

(2) Una altra opció de millora a realitzar podria ser tenir una altra definició en el diagrama conceptual de dades. Per aconseguir aquest propòsit podem tenir una organització diferent en les classes :

- Classes per emmagatzemar les dades (matrius amb dades)

- Classes per realitzar operacions que actuen sobre aquestes dades

A considerar,

- Classe Atribut per definir regla.

- Classe per als Atributs Discretitzats, per poder heredar d'atribut i modificar si cal.

- Llista de regles, on cada cada regla tindria : antecedent, consequent, suport, confiança, etc.

(3) Calcular els Conjunts freqüents de mida K:

També volem considerar una optimització a l'algorisme en qüestió pel qual necessitem una millora en la distribució de les classes: Trobar primer els conjunts d'atributs ε-Freqüents i a partir d'aquests inferir les regles corresponents. Això a part d'una millora en eficiència ens permetría eliminar casos repetits que tenim en la implementació actual, a més de facilitar el càlcul dels test d'exemples.

(4) En Algorisme:

Per altra banda, un cop solucionat el problema del càlcul de conjunts, això ens permetrà poder fer millores en l'eficiència de l'algorisme, i podem considerar-lo amb una execució de forma iterativa.

(5) Per executar test de proves:

Seria més convenient tenir un Main alternatiu exclusivament per testejar de manera que es pugui,

- crear llista
- generar regles
- indicar confiança

i fer-lo funcionar com si del usuari es tractés, de manera que llegeix del fitxer i generi resultats.

Aquest mecanisme ajudarà, en canvis que es produeixin en Generar Regles, a la comprovació que funciona de la mateixa manera que abans de realitzar aquest canvis. Assegurant la correcta execució.