DATA MINING

Projectes de programació
Q2 2019-2020

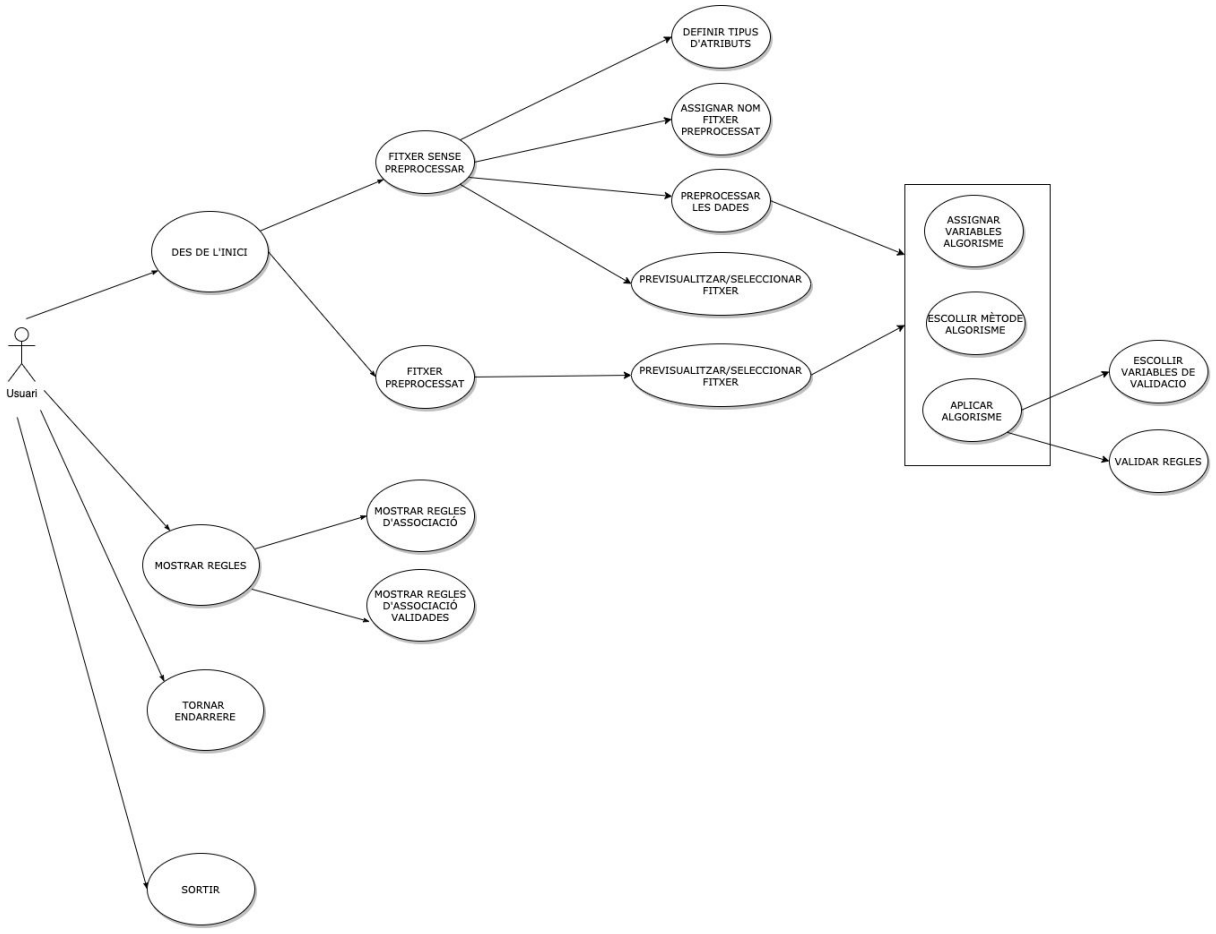
GRUP 21, SUBGRUP 4.2

Oriol Catalán - oriol.catalan.fernandez
Alex Rubiano - alex.rubiano
Aleix Vila - aleix.vila.berenguel
Lliurament 1.0

ÍNDEX

1. CASOS D'ÚS

1.1. Diagrama de casos d'ús



Es pot trobar una imatge del diagrama de casos d'ús a la carpeta DOC.

1.2. Descripció dels casos d'ús

1.2.1. Des de l'inici:

- Descripció: Quan l'usuari entra al programa i decideix començar un nou procés de data mining per a treure unes regles d'associació d'un fitxer csv. Es passarà del menu principal a un menu per a escollir el tipus de fitxer que s'escollirà (a preprocessar o ja preprocessat amb antel·lació).

1.2.2. Fitxer sense preprocessar:

- Descripció: L'usuari vol realitzar el procés de data mining a partir d'un fitxer que encara no ha estat preprocessat. Es passarà del menu per escollir el tipus de fitxer al menú amb les opcions corresponents a un fitxer sense preprocessar.

1.2.3. Fitxer Preprocessat:

- Descripció: L'usuari vol realitzar el procés de data mining a partir d'un fitxer que ja ha estat preprocessat (no cal que sigui amb el mateix programa). Es passarà del menu per escollir el tipus de fitxer al menú amb les opcions corresponents a un fitxer preprocessat.

1.2.4. Previsualitzar/Seleccionar Fitxer :

- Descripció: L'usuari vol escollir el fitxer que farà servir per el data mining, el qual abans es previsualtzarà de cara a que l'usuari pugui veure que està escollint un fitxer adequat. Aquest cas d'ús estarà present tant pels fitxers sense preprocessar com pels que ja han estat preprocessats prèviament.

1.2.5. Definir tipus atributs:

- Descripció: L'usuari escull per cada atribut corresponent al fitxer, el seu tipus. D'aquesta forma l'usuari podrà decidir com es discretitzaran els atributs, per d'aquesta forma poder aconseguir les regles que s'adjustin més a les seves necessitats. Si s'escull malament un tipus d'atribut, el programa finalitzarà.

1.2.6. Assignar nom fitxer preprocessat:

- Descripció: L'usuari posa ,abans del preprocessat del fitxer inicial, el nom del fitxer un cop ja preprocessat, el qual es guardarà en una carpeta específica, encarregada tots els fitxers preprocessats.

1.2.7. Preprocessar les dades:

- Descripció: L'usuari passa al preprocessat del seu fitxer a partir dels tipus d'atributs definits prèviament. El fitxer es guarda a la carpeta anomenada prèviament i es passa a la finestra d'algorisme.

1.2.8. Assignar variables algorisme

- Descripció: L'usuari escull per el procés de data mining quins seran els seus llindars mínims de suport i confiança i el seu llindar màxim de número de regles. A més a més també escull quin serà el nom del fitxer a on es guardaran les regles extretes a partir del procés. Aquest fitxer serà guardat en una carpeta específica per tots els fitxers d'aquest tipus, i que contenen el model de regles d'associació extret d'un fitxer .csv.

1.2.9. Escollir mètode algorisme

- Descripció: L'usuari escull entre dos mètodes per a realitzar el seu data mining:
    - El primer consisteix en partir el fitxer preprocessat en dos (80% i 20%), per d'aquesta forma poder validar posteriorment les regles extretes del 80% amb el 20%.
    - El segon consisteix en directament, passar-li el 100% del fitxer per a observar directament les regles extretes amb la totalitat del fitxer.

1.2.10. Aplicar algorisme

- Descripció: L'usuari, apartir dels paràmetres definits anteriorment, decideix començar el procés de data mining per a poder extreure les regles d'associacions corresponents. Es guarda el fitxer amb les regles a la carpeta específica anomenada anteriorment i es passa de la finestra d'algorisme a la de validació si s'ha escollit el mètode 1 l'algorisme sobre escull el nom del

fitxer que guarda les regles d'associació i es mostren les regles corresponents.

1.2.11. Escollir variables de validació:

- Descripció: L'usuari escull els marges percentuals de diferència entre els suports i confiances de les regles extretes(80% del fitxer) i dels registres restants (20% del fitxer) i escull el nom del fitxer a on es guardaren les regles un cop validades. Aquest fitxer anirà a parar a una carpeta específica a on es guardaran tots els fitxers que contenen regles ja validades.

1.2.12. Validar regles:

- Descripció: L'usuari decideix validar les regles a partir dels paràmetres definits anteriorment. Un cop realitzat el procés de validació, es mostren les regles que són vàlides i es guarden en el fitxer amb nom assignat per l'usuari anteriorment.

1.2.13. Mostrar regles:

- Descripció: Quan l'usuari vol consultar regles d'associació ja trobades anteriorment (ja siguin validades o sense validar). Es passarà del menu principal a la vista que mostrarà les regles.

1.2.14. Mostrar regles d'associació:

- Descripció: L'usuari decideix que vol consultar unes regles sense validar, i agafa un dels fitxers guardats a la carpeta específica que conté fitxers amb regles que no han estat validades.

1.2.15. Mostrar regles d'associació validades:

- Descripció: L'usuari decideix que vol consultar unes regles ja validades, i agafa un dels fitxers guardats a la carpeta específica que conté fitxers amb regles que ja han estat validades.

1.2.16. Tornar endarrere:

- Descripció: L'usuari decideix que vol tornar endarrere ja que ha realitzat un error. Es tornarà a la vista anterior.

1.2.17. Sortir:

- Descripció: L'usuari decideix acabar l'execució del programa i pitja el botó sortir de qualsevol de les vistes.

Aquests casos d'ús són el que farem servir per a la versió final del projecte. Aquí estan encapsulades totes les funcionalitat del nostre sistema.

2. MODEL CONCEPTUAL DE LES DADES

2.1. Diagrama UML

Hem adjuntat el diagrama complet en un pdf en la carpeta de documentació, ja que pel format i el mida no es veurien correctament en aquest document.

2.2. Especificació de cada classe

2.2.1. Dades

- **Descripció:** Classe que representa els diferents tipus de dades que es troben durant el procés de mineria de dades.

- **Atributs:**
  - String *ruta*: Ruta del fitxer escollit.
  - Object[][] *matriuDades*: Matriu amb els registres del fitxer.
  - Object[][] *matriuDadesInvertides*: MatriuDades invertida.
  - ArrayList<ArrayList<Item>> *matriuDiscretitzada*: Dades ja discretitzades.
  - ArrayList<ArrayList<Item>> *matriuAlgorisme*: Dades que li passes a l'algorisme.
  - ArrayList *tipusAtributs*: Llista amb el tipus de tots els atributs.
  - ArrayList *nomAtributs*: Llista amb els noms de tots els atributs.

- **Explicació:** Aquesta classe conté totes les variables que usem de les dades del fitxer (ruta, noms i tipus d'atributs, registres, etc) que volem preprocessar i/o treure les regles corresponents. D'aquesta forma, tant les seves subdades com la classe algorisme, podran usar-les en qualsevol moment per a realitzar els seus mètodes.

2.2.2. DadesBrut

- **Descripció:** Subclasse de la classe Dades, que representa les dades sense cap mena de preprocès.

- **Atributs:**
    - Al ser una subclasse de la superclasse Dades, té els atributs de la superclasse també.

A més a més també usa aquestes variables per a realitzar les seves funcionalitats.

- File *arxiu*: Arxiu escollit.
- FileReader *fitxer*: Fitxer lectura.
- BufferedReader *buffer*: Emmagatzemador de lectura de fitxer.
- ArrayList *filesTotals*: Llista de les files del fitxer.
- Int *columnes*: Variable contenidora del numero de columnes del fitxer.
- ArrayList *tipusAtributs*: Conté el tipus de cada atribut del fitxer.

- **Explicació:** És un dels tipus de Dades que tenim, el qual conforma les dades que encara no han estat discretitzades.
  Aquesta classe llegeix el fitxer csv inicial que l'usuari ha escollit inicialment, i es guarda els valors de cada registre en el format correcte depenent de l'atribut al que correspongui, agafant l'arraylist de String **tipusAtributs**, que ha estat prèviament emplenada per l'usuari al demanar-li quin és el tipus de cada atribut (l'usuari ha d'introduir per pantalla de quin tipus és cada atribut del fitxer). Els corresponents registres són emmagatzemats a les matrius d'objectes **matriuDades** i **matriuDadesInvertides** per a poder discretitzar aquests posteriorment.

2.2.3. DadesPreprocessades

- **Descripció:** Subclasse de la classe Dades que representa les dades un cop discretitzades.

- **Atributs:**
    - Al ser una subclasse de la superclasse Dades, té els atributs de la superclasse també.

A més a més també usa aquestes variables per a realitzar les seves funcionalitats.

- File *arxiu*: Arxiu escollit (preprocessat)
- FileReader *fitxerL*: Fitxer lectura.
- BufferedReader *buffer*: Emmagatzemador de lectura de fitxer.
- ArrayList *filesTotals*: Llista de files del fitxer.
- FileWriter *fitxerE*: Fitxer escriptura.
- PrintWriter *escriptor*: Escriptor al fitxer.
- String[][] *matriuJaPreprocessada*: Matriu del fitxer preprocessat.

- **Explicació:** És un dels tipus de Dades que tenim, el qual conforma les dades que han estat ja discretitzades.

  Aquesta classe té un ampli ventall de funcionalitats, ja que primerament, si l'usuari escull un fitxer que ja ha estat preprocessat anteriorment, la classe s'encarregarà de llegir el fitxer i emmagatzemar tots els registres en format item a l'arraylist d'arraylist de Item **matriuDiscretitzada**, o també, si l'usuari ha escollit un fitxer sense preprocessar, la classe agafarà les matrius **matriuDades** i **matriuDadesInvertides (**que han estat emplenades al pas previ a DadesBrut) i discretitzarà tots els seus atributs numèrics, emmagatzemant-los posteriorment amb els atributs de tipus no numèric (els quals no es discretitzen) a **matriuDiscretitzada**, la qual guarda tots els registres ja discretitzats en format String.

2.2.4. DadesModificades

- **Descripció:** Subclasse de la classe Dades que representa les dades un cop discretitzades, inspeccionades i dividides.
- **Atributs:**
    - Al ser una subclasse de la superclasse Dades, té els atributs de la superclasse també.

A més a més també usa aquestes variables per a realitzar les seves funcionalitats.

- ArrayList<ArrayList<Item>> *matriuValidacio*: Dades que s'usen per a validar regles d'associació.

- **Explicació:** És un dels tipus de Dades que tenim, el qual conforma les dades que han estat ja discretitzades i separades en dos per a poder aplicar l'algorisme i validar les regles d'associació corresponents.
  Aquesta classe s'encarrega d'agafar la **matriuDiscretitzada** per a dividir-la en dos arraylist d'arraylist de String les quals són **matriuAlgorisme** i **matriuValidacio.** Aquesta divisió es realitzarà agafant el 20% dels registres aleatòriament i guardant-los a **matriuValidacio,** per finalment guardar el 80% restant a **matriuAlgorisme**. Aquesta ultima matriu el la utilitzarem per obtenir les regles d'associació aplicant el algorisme Apriori.

2.2.5. Algorisme

- **Descripció:** Classe implementada que representa l'algorisme Apriori, el qual usarem per extreure els conjunts freqüents del conjunt de dades i posteriorment les regles d'associació corresponents.
- **Atributs:**
    - Hashtable<Integer, ArrayList<Item>> *antecedent*: HashTable amb els antecedents de les diferents regles d'associació trobades.
    - Hashtable<Integer, ArrayList<Item>> *consequent*: HashTable amb els conseqüents de les diferents regles d'associació trobades.
    - Hashtable<Integer, Double> *llistaSuport*: HashTable amb els suports de les diferents regles d'associació trobades.
    - Hashtable<Integer, Double> *llistaConfianca*: HashTable amb les confiances de les diferents regles d'associació trobades.

- double *suport*: Llindar mínim de suport.
- double *confiancaMinima*: Llindar mínim de confiança.
- int *maxRegles*: màxim nombre de regles d'associació.
- ArrayList<ArrayList<Item>> *dadesModificades*: Llista del 80% dels registres del fitxer inicial.
- Dades dades: Dades a les que se li aplica l'algorisme.

- **Explicació:** És la classe que aplica els algorismes per treure les regles s'associació a partir de les dades d'entrada discretitzades. Primerament aplica l'algorisme Apriori on es treuen els conjunts freqüents, és a dir, els conjunts amb màxim nombre d'items que tenen un suport igual o superior a **suport**, i això es guarda a **ConjuntsFrequents**. Després, amb la llista de conjunts freqünts aplica l'algorisme a cada conjunt per treure les regles d'associació que tinguin una confiança igual o superior a **confiancaMinima**. Per cada regla d'associació l'antecedent es guarda a **antecedent**, el conseqüent es guarda a **consequent**, el suport a **llistaSuport** i la confiança a **llistaConfianca**, i tots amb la mateixa clau. A més, si el nombre de regles d'associació supera **maxRegles**, es redueix el nombre de regles per tenir les **maxRegles** regles d'associació amb major confiança.

2.2.6. Validació

- **Descripció:** Classe associativa que representa la validació d'un model de regles d'associació sobre un conjunt de dades diferent de l'escollit inicialment.
- **Atributs:**
  - ArrayList<ArrayList<Item>> *matriuValidacio*: Llista del 20% dels registres del fitxer inicial que no han estat passades a l'algorisme.
  - Hashtable<Integer, ArrayList<Item>> *antecedent*: HashTable amb els antecedents de les diferents regles d'associació trobades.
  - Hashtable<Integer, ArrayList<Item>> *consequent*: HashTable amb els conseqüents de les diferents regles d'associació trobades.
  - Hashtable<Integer, Double> *suport*: HashTable amb els suports de les diferents regles d'associació trobades.

- Hashtable<Integer, Double> *confianca*: HashTable amb les confiances de les diferents regles d'associació trobades.
- Double *MargeSup*: Double amb el valor del marge de suport que donem per a que la regla pugui ser acceptada en la validació.
- Double *MargeConfi*: Double amb el valor del marge de confiança que donem per a que la regla pugui ser acceptada en la validació.
- Algorisme *algo*: Algorisme del qual hem agafat les regles a través dels hashtables.
- DadesModificades *DadesMod*: Dades de les que hem agafat matriu Validació.

- **Explicació:** Aquesta classe s'encarrega de validar les regles d'associació trobades a l'algorisme aplicant-les al 20% restant dels registres que com s'ha comentat abans estan guardats a **matriuValidacio**. Anirà agafant l'antecedent i conseqüent de cada regla i recorrerà tots els registres de **matriuValidacio** per treure el suport i confiança de cadascuna respecte a aquestes dades. A aquest suport i confiança, l'usuari ha afegit un marge per a que els valors del registres de la **matriuValidacio** siguin una mica més flexibles a la hora de validar les regles. Finalment, les regles que l'usuari ha decidit que són vàlides, apareixeran per pantalla.

2.2.7. Item

- **Descripció:** Classe que identifica cada valor d'un registre amb el nom de l'atribut al qual corresponen i el valor.
- **Atributs:**
  - String *id*: Nom de l'atribut al qual pertany l'ítem.
  - String *valor*: Valor de l'ítem.
- **Explicació:** Fa la funció d'una tupla on poder guardar el valor de l'item i el nom de l'atribut al qual pertany. A més té una funció pública per comparar dos objectes d'aquesta classe i veure si són iguals.

2.2.8. ItemSet

- **Descripció:** Classe que serveix per a identificar els itemsets del fitxer amb el suport corresponent a cadascun.
- **Atributs:**
    - ArrayList<Item> *set*: Llista d'ítems d'una fila del fitxer.
    - double *suport*: suport de l'itemset corresponent.
- **Explicació:** Fa la funció d'un conjunt d'items i funciona com una tupla, on es guarda la llista d'items del conjunt i el valor del suport del conjunt a les dades.

2.2.9. CtrlDomain

- **Descripció:** Classe controlador de la capa de domini
- **Atributs:**
    - Dades *Dades*: Dades del fitxer amb el qual interactuar.
    - DadesBrut *DadesBrut*: Dades abans de discretitzar del fitxer.
    - DadesPreprocessades *DadesPreprocessades*: Dades ja discretitzades del fitxer.
    - DadesModificades *DadesModificades*: Dades ja discretitzades separades al 80/20 per a aplicar l'algorisme i validar les regles
    - Algorisme *Algorisme*: Algorisme i regles d'associació corresponents.
    - Validacio *Validacio*: Validació de les regles que extreu la classe algorisme.
    - String *text*: Variable que usem per la comunicació amb la capa de presentació.
- **Explicació:** El controlador de la capa de domini. Ell s'encarrega de comunicar-se amb les altres capes del sistema i de rebre peticions d'altres capes. Aquestes peticions les va dirigint a les classes que li pertoquin. Aquesta classe també és útil per relacionar totes les classes de la capa de Domini, i que no hi hagi molt embolic amb les relacions entre elles.

2.2.10. CtrlDatabase

- **Descripció:** Classe controlador de la capa de dades.

- **Atributs:**.

- DatabaseF *IO*: Classe que agafa el path del fitxer escollit.

- **Explicació:** Classe que s'encarrega de la comunicació amb la capa de domini i de dirigir les peticions al lloc corresponent.

2.2.11. DatabaseF

- **Descripció:** Classe de la capa de dades per a treure els paths dels fitxers a usar.
- **Explicació:** Aquesta classe s'encarrega de retornar els diferents paths dels fitxers que va escollint l'usuari.

2.2.12. CtrlPresentation

- **Descripció:** Classe controlador de la capa de presentació.
- **Atributs:**
  - CtrlDomain *ctrlDomain*: Instancia del controlador de domini.
  - viewMenuPrincipal *viewMP*: Instancia de la vista del menu principal.
  - viewMenuFitxers *viewMF*: Instancia de la vista del menu principal.
  - viewMenuMostrarRegles *viewMSR*: Instancia de la vista del menu principal.
  - viewMenuFitxersSensePre *viewMFSP*: Instancia de la vista del menu principal.
  - viewMenuFitxersPre *viewMFP*: Instancia de la vista del menu principal.
  - viewValidacioRegles *viewVR*: Instancia de la vista del menu principal.
  - viewAlgorisme *viewA*: Instancia de la vista del menu principal.
  - viewImprimirReglesValidades *viewIRV*: Instancia de la vista del menu principal.
  - ArrayList *tipusAtributs*: Arraylist amb el tipus dels atributs.

- **Explicació:** El controlador de la capa de presentació. S'encarrega de comunicar-se amb la capa de domini a través del controlador de domini, i li passa les peticions i dades de les diferents vistes cap a les classes que s'encarreguen de fer les operacions pertinents.

2.2.13. Vistes: objectes i funcions comunes

- **Descripció:** Aquí expliquem els objectes comuns que hi ha a moltes de les vistes.
- **Atributs:**
  - JButton *exitButton*: Botó que tanca el programa.
  - JButton *backButton*: Botó que et porta a la vista anterior.
  - JPanel *menuPanel*: Objecte que conté tots els elements de la vista.
  - CtrlPresentacio *ctrlPresentacio*: Instancia del controlador de presentacio.
  - Int *xMouse*: Variable que conté la posició del mouse en el eix de les X.
  - Int *yMouse*: Variable que conté la posició del mouse en el eix de les Y.
  - String *path*: String que conté el path cap al fitxer que hem d'utilitzar.

- **Explicació:** Aquí tenim els objectes que son comuns a totes o a la majoria de les vistes que utilitzem. També son comuns els metodes de fer visible i invisible la vista, el de inicialitzar els components i la constructora, que té una estructura semblant per a totes.

2.2.14. Vista Algorisme

- **Descripció:** La vista que s'encarrega de llençar el algorisme Apriori.
- **Atributs:**
  - JButton *AlgButton*: Botó que executa el algorisme.
  - JTextField *SuptextField, ConftextField, NumRtextField, FitxerDtextField*: Són camps que el usuari omple per donar valor a les variables del algorisme.
  - JLabel *NumRLabel, ConfLabel, SupLabel, FitxerDLabel*: Labels amb un text descriptiu de cada camp.
  - JRadioButton *NoValidacioRadioButton,ValidacioRadioButton*: Dos botons complementaris que permeten executar el algorisme amb el 80% de les dades per fer la posterior validació, o obviar la validacio de les regles.

- **Explicació:** En aquesta vista rebem els paràmetres que introdueix l'usuari per aplicar el algorisme.

2.2.15. Vista Imprimir Regles Validades

- **Descripció:** La vista que s'encarrega d'imprimir els les regles d'associació obtingudes en la validació.
- **Atributs:**
    - JTextArea *textArea*: Area de text que conté el contingut del fitxer.
    - JButton menuButton: Botó per anar al menú principal.
    - String path : path del fitxer amb les regles validades.

- **Explicació:** En aquesta vista agafem el fitxer amb path igual a la variable path per a imprimir les regles d'associació validades a través del textArea. D'aquesta manera, l'usuari pot veure el resultat final del procés de data mining realitzat.

2.2.16. Vista Menu Assignar Tipus Atributs

- **Descripció:** La vista que s'encarrega de assignar el tipus dels atributs.
- **Atributs:**
    - JButton *OKButton*: Botó que confirma la operació.
    - JComboBox *comboBox*: Menu que conté tots els possibles tipus d'atributs.

- **Explicació:** En aquesta vista de tipus "pop-up" utilitzem el menu **comboBox** per donar al usuari 4 possibles tipus d'atributs amb que processarem les dades, on confirmarem cadascun per separat mitjançant el **OKButton**.

2.2.17. Vista Menu Fitxers

- **Descripció:** La vista que s'encarrega de dirigir al usuari a preprocessar el fitxer inicial o utilitzar un fitxer ja preprocessat.

2.2.18. Vista Menu Fitxers Pre

- **Descripció:** La vista que s'encarrega d'agafar el fitxer preprocessat.
- **Atributs:**
    - JButton *AlgButton*: Botó que dirigeix al usuari a la vista del algorisme.
    - JButton *getFileButton*: Botó que permet al usuari agafar el fitxer d'un directori.
    - JTextArea *textArea*: Area de text que conté el contingut del fitxer.
    - JFileChooser *agafaFitxer:* Objecte que permet agafar el fitxer.

- **Explicació:** En aquesta vista l'usuari tria el fitxer amb el que vol treballar, sent aquest un fitxer ja preprocessat. Aquesta vista permet al usuari també poder previsualitzar aquest arxiu per comprovar que es efectivament el que vol i no s'ha equivocat, i amb el **AlgButton** et porta a la vista d'algorisme per poder seguir amb la execució.

2.2.19. Vista Menu Fitxers Sense Pre

- **Descripció:** La vista que s'encarrega d'agafar el fitxer inicial.
- **Atributs:**
    - JButton *PrePButton*: Botó que dirigeix al usuari a la vista del algorisme, alhora que fa que el sistema preprocessi el fitxer.
    - JButton *selectFButton*: Botó que permet al usuari agafar el fitxer d'un directori.
    - JTextArea *textArea*: Area de text que conté el contingut del fitxer.
    - JFileChooser *agafaFitxer:* Objecte que permet agafar el fitxer.
    - JButton *AssignaATRButton*: Botó que obre els "pop-up"s per assignar els atributs.

- JTextField *textField*: Camp de text per donar nom al fitxer preprocessat.

- **Explicació:** En aquesta vista l'usuari tria el fitxer inicial amb el que vol treballar. Aquesta vista permet al usuari també poder previsualitzar aquest arxiu per comprovar que es efectivament el que vol i no s'ha equivocat. L'usuari seguidament ha d'assignar un valor al fitxer preprocessat, emplenant el camp **textField**, i seguidament utilitzar **AssignaATRButton** per anar amb l'assignació d'atributs.Finalment, amb el **PrePButton** el sistema et porta a la vista d'algorisme (havent preprocessat el fitxer) per poder seguir amb la execució.

2.2.20. Vista Menu Mostrar Regles

- **Descripció:** La vista que s'encarrega de mostrar els fitxers amb les regles d'associació.
- **Atributs:**
  - JButton *selectFButton*: Botó que permet al usuari agafar el fitxer d'un directori.
  - JTextArea *textArea*: Area de text que conté el contingut del fitxer.
  - JFileChooser *agafaFitxer:* Objecte que permet agafar el fitxer.

- **Explicació:** En aquesta vista l'usuari tria el fitxer que conté regles d'associació, tant les regles d'algorisme com les regles validades, i pot visualitzar tot el contingut del fitxer.

2.2.21. Vista Menu Principal

- **Descripció:** La vista que s'encarrega de dirigir al usuari a començar l'execució del programa o a mostrar els fitxers amb les regles d'associació.

2.2.22. Vista Validacio Regles

- **Descripció:** La vista que s'encarrega de validar les regles d'associació.
- **Atributs:**

- JButton *VALIDARButton*: Botó que executa la validació.
- JTextField *textFieldConfi, textFieldSup, textField1*: Són camps que el usuari omple per donar valor als marges de la validació, i també el nom del fitxer amb les regles validades.
- JLabel *labelConfi, labelSup, labelNom*: Labels amb un text descriptiu de cada camp.

- **Explicació:** En aquesta vista l'usuari dona valors als marges per al validació i també el nom del fitxer que obtindrà, i fent servir el **VALIDARButton**, activa la validació de les regles d'associació i es guarda les regles validades a un fitxer amb el nom assignat.

3. DESCRIPCIÓ DE LES ESTRUCTURES DE DADES I ALGORISMES USADES

3.1 Apriori

3.1.1 Estructures de dades

- Item: Classe que ens permet representar un item. S'utilitza com una tupla amb un String pel valor i altre String per l'identificador de l'atribut de la transacció a la qual pertany.
- ItemSet: Classe que ens permet representar un conjunt de items. S'utilitza com una tupla amb un ArrayList<Item> per llistar els items del conjunt i amb un double per guardar el suport del conjunt.
- ArrayList<ItemSet>: implementa la llista de conjunts freqüents.

3.1.2 Algorisme

L'algorisme Apriori és un algorisme de generació de candidats. L'objectiu d'aquest algorisme és que a partir del conjunt de transaccions discretitzades i donat un llindar anomenat Suport, es generin els conjunts amb major número d'items que apareguin en les transaccions amb una proporció igual o superior a Suport.

L'algorisme primer genera el conjunt "items" que són tots els items diferents que hi ha en el conjunt de transaccions. Després l'algorisme treballa amb un bucle iteratiu on per cada iteració la variable "nombreItems" s'incrementa en 1 (comença amb 1). En cada iteració d'aquest bucle primer es genera una llista de totes les possibles permutacions del conjunt "items" amb el número "nombreItems" de items; segon, per cada subconjunt d'items sortit de la permutació es recorre el conjunt de transaccions d'entrada i compta les vegades que apareix i s'extreu el suport, si aquest suport és major o igual al mínim establert el subconjunt es guarda en "itemSetsAmbSuportMinim"; tercer, si "itemSetsAmbSuportMinim" és buit, vol dir que no hi ha subconjunts de "nombreItems" que arribin al suport mínim, per tant es retorna la llista de conjunts d'items freqüents de l'anterior iteració.

3.2 Regles d'associació

3.2.1 Estructures de dades

- Hashtable<K, V>: implementa les regles d'associació en 4 hashtables, un pel conjunt d'antecedent, un pel conjunt de conseqüent, un pel suport i altre per la confiança, tots sincronitzats amb una mateixa clau per cada regla.

3.2.2. Algorisme

L'algorisme "regles_associacio" genera les regles d'associació que superen o igualen la confiança mínima a partir d'un conjunt d'items freqüent extret de l'algorisme Apriori.

L'algorisme treballa amb un bucle iteratiu on per cada iteració s'incrementa "numElem" en 1 i funciona mentre sigui menor que el número d'items del conjunt freqüent. En cada iteració es genera una llista amb tots els possibles subconjunts amb "numElem" d'items del conjunt freqüent. Després, per cada subconjunt es compta les vegades que apareix en el conjunt de transaccions i es calcula la confiança. Si aquesta confiança és major o igual a la confiança mínima el subconjunt s'estableix com a antecedent d'una regla d'associació, els items disjunts s'estableixen com a conseqüent de la regla i es guarda el seu suport i confiança.

3.3 Reduir regles

3.3.1 Estructures de dades

- Hashtable<K, V>>: implementa les regles d'associació en 4 hashtable, un pel conjunt d'antecedent, un pel conjunt de conseqüent, un pel suport i altre per la confiança, tots sincronitzats amb una mateixa clau per cada regla.

3.3.2 Algorisme

L'algorisme "reduir_regles" redueix les regles originals fins a tenir el màxim nombre de regles establert per l'usuari amb les màximes confiances possibles.

L'algorisme rep per paràmetre els 4 HashTable que representen les regles d'associacions i es guarda, a una varible local, la mida de un dels HashTable per saber el nombre de regles originals. Es fa un bucle iteratiu amb màxim de regles iteracions, on per cada iteració s'agafa la regla amb major confiança i es guarda a les estructures de les regles que estan com a variables globals, i es borra la regla de les estructures originals.

4. EXPLICACIÓ SOBRE EL TEST UNITARI DE LES CLASSES DADES MODIFICADES I ALGORISME

En aquest test unitari JUnit, s'ha comprovat que les classes **DadesModificades** i **Algorisme** funcionen de forma correcta a partir d'aplicar-hi una serie de tests cadascuna a través d'un test case, a on s'esperava un resultat i aquestes han hagut de respondre de la forma esperada. D'aquesta manera, hem pogut anar veient els possibles errors que apareixien en el nostre i els hem pogut canviar d'una forma més ràpida i sencilla.

En quant als **atributs** usats en el test hem usat aquests:

- ArrayList<ArrayList<Item>> matriuTest = new ArrayList<>() : En aquest atribut s'ha emmagatzemat tots els registres pertanyents al fitxer amb path igual a l'atribut rutaEntrada.
- ArrayList<Item> filesTotals = new ArrayList<>(): atribut que representa una de les files del fitxer amb path igual a l'atribut rutaEntrada.
- rutaEntrada = (CtrlDomain.*loadFileTest*() + "provaTest.csv") : Path del fitxer que s'usarà per tot el test.
- DadesModificades instanceDadesMod: Instància de DadesModificades per a poder realitzar el test de la classe.
- Algorisme instanceAlgorisme: Instància d'Algorisme per a poder realitzar el test de la classe.

En quant al que s'ha fet abans de passar als tests **(@before)**, trobem aixo:

@Before

public void setUp(): funció que crea una instància de DadesModificades, extreu la MatriuTest del fitxer amb path = rutaEntrada i corre l'execució de la classe DadesModificades per a poder tenir els resultats a les variables amb els que realitzarem el test.

public void TestDadesModificades(): Aquesta funció realitza la tasca d'extreure la MatriuTest del fitxer amb path = rutaEntrada. Aquesta funció es cridada per SetUp.

En quant als tests realitzades a les classes trobem aquests:

- DadesModificades:

@Test

public void DivisioMatriuCorrecte (): Comprova que la partició 80/20 de la matriuTest per a passar el 80% a l'algorisme i guardar el 20% per validar les regles posteriorment és correcte.

- Algorisme:

@Test

public void Algorisme_Suport_Confiança_MaxRegles_Correcte1(): Comprova que l'algorisme extreu les regles seguint els llindars establerts de Suport i Confiança (els quals són llindars mínims) i el de númeroMàximDeRegles (llindar màxim) i que no hi ha cap regla que contingui un suport o confiança per sota dels llindars o que surten més regles del màxim demanat.

@Test

public void Algorisme_Suport_Confiança_MaxRegles_Correcte2(): Realitza el mateix que el test anterior, però amb valors diferents, d'aquesta manera, ens assegurem que l'algorisme funcionarà amb valors diferents.

@Test

public void HashTablesSincronitzats(): Comprova que els hashtables que comformen els antecedents, conseqüents, suports i confiances de les regles extretes tenen la mateixa mida, d'aquesta manera, ens assegurem de que l'algorisme treu les regles emmagatzemant tots els components d'aquestes de forma correcte.

D'aquesta manera, hem pogut comprovar el bon funcionament d'aquestes dues classes, en especial la classe Algorisme, que és la més important de tot el projecte.

5. CLASSES IMPLEMENTADES PER CADA COMPONENT DEL GRUP

Capa de Domini i Capa de Dades:

Les classes **Dades, DadesBrut i DadesPreprocessades** com el controlador **Ctrl Database** amb la seva classe **DatabaseF** han estat implementades per ***Oriol Catalán.***

Les classes **Itemset, Main Driver i Algorisme** han estat implementades per ***Alex Rubiano.***

Les classes **Item, DadesModificades, Validacio,** com el Controlador **CtrlDomain** i el test **TestDadesModificades_Algorisme** han estat implementades per ***Aleix Vila.***

Capa de Presentació:

Les classes **viewImprimirReglesValidades, viewValidacioRegles, viewMenuAsignarTipusAtributs i viewMenuMostrarRegles** han estat implementades per ***Oriol Catalán.***

Les classes **viewMenuFitxersSensePre, viewMenuFitxersPre i la millora de la classe Algorisme** han estat implementades per ***Alex Rubiano.***

Les classes **CtrlPresentation, viewMenuPrincipal, viewMenuFitxers, viewAlgorisme i el canvi de la classe validació** han estat implementades per ***Aleix Vila.***