SISTEMA MULTIAGENT PER A LA CLASSIFICACIÓ DE DADES

Estudiant:
OSCAR UJAQUE PEREZ
Grau Enginyeria Informàtica

Consultor:
Dr. DAVID ISERN ALARCÓN

23/12/2014

ÍNDEX

ÍNDEX DE FIGURES

0. Resum

En aquest treball es pretén dissenyar un sistema multiagent que, amb l'aplicació de diferents algorismes supervisats de classificació, presenti a l'usuari, donades unes dades, la classificació més probable.

El projecte es dissenya, mitjançant l'arquitectura Jade (en llenguatge de programació Java). En aquest, diferents agents amb diferents comportaments i amb l'ús d'una ontologia per entendre's interactuaran per analitzar dades i extreure'n conclusions.

El conjunt de dades que avalua en aquest projecte es referent al tipus d'accessibilitat de diferents vehicles. Així doncs, atès uns atributs de cada vehicle (preu, manteniment, nombre de portes, capacitat en persones, volum del portaequipatges i tipus de seguretat) es prediu el tipus d'accessibilitat compresa entre (molt bona, bona, accessible o no accessible).

En l'ús del programari, un usuari manipula un agent principal (o Manager). Aquest agent principal envia dades a classificar a diversos agents classificadors (distribuïts o no en l'espai). Aquests agents classificadors, mitjançant algorismes supervisats, fan una predicció de les dades rebudes. Un cop fet el càlcul, s'envien les classificacions a l'agent principal perquè, amb totes les prediccions rebudes de tots els agents classificadors, calculi la que tingui una probabilitat més alta.

La següent imatge mostra a nivell il·lustratiu el funcionament del sistema.

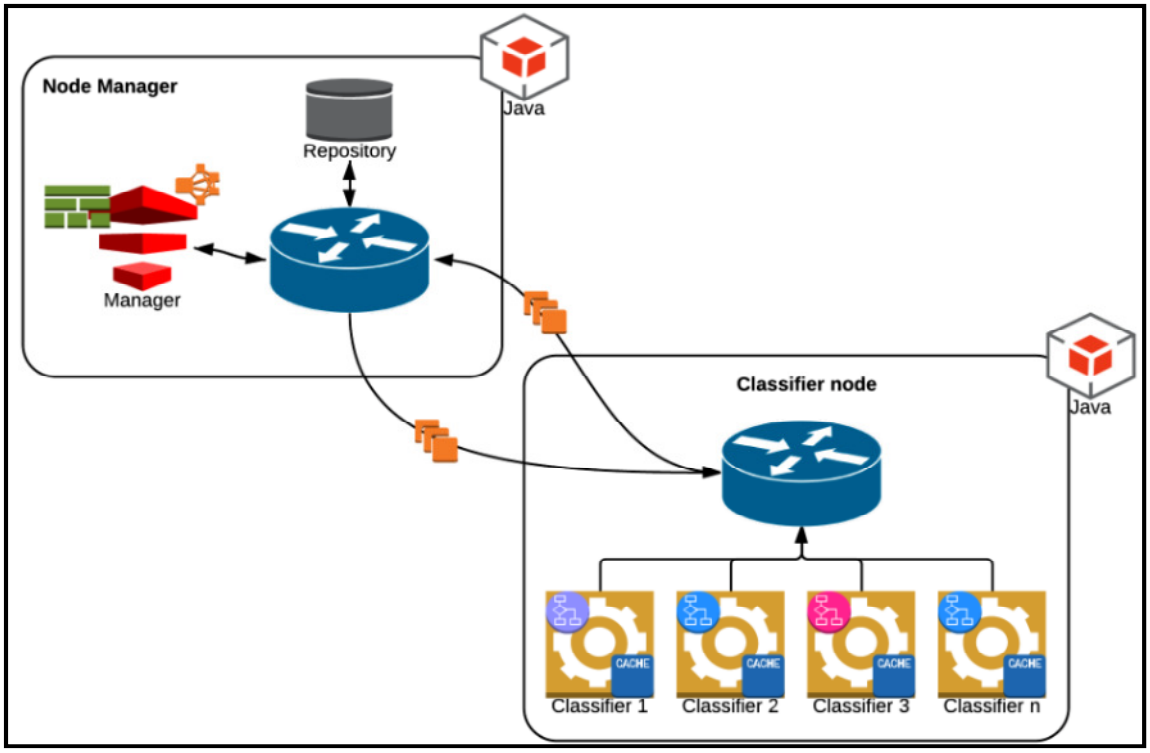

Figura 1: Interacció Agents

1. INTRODUCCIÓ

En certs àmbits, com la medicina o l'empresa privada, volem extraure informació addicional a conjunts de dades que estan emmagatzemades en diversos servidors. Però, molts cops, aquest servidors, a més d'estar distribuïts, són privats, el que implica que sigui impossible consultar-ne les dades.

Per tal de poder accedir a aquesta informació privada d'alguna forma, es pot configurar un sistema composat de diversos agents, que, a partir d'algorismes de predicció supervisats, extreguin informació de cada servidor determinat i no en violin la privadesa.

Per tant, amb un sistema com l'anterior, podem extreure informació de cada servidor, mitjançant un agent classificador que, a posteriori, es pot enviar a un agent principal perquè la tracti i arribi a algun estat o conclusió final. D'aquesta forma, consultem la informació que volem de qualsevol servidor i no en violem la privadesa.

A més, aquesta metodologia, com que el volum de dades és molt elevat (hi pot haver molts servidors als que consultar), implica que el resultat final obtingut sigui bastant precís.

Alguns projectes ja s'ha dissenyat amb aquesta configuració. Per exemple, el projecte *HealthAgents1*, que mitjançant agents distribuïts computa informació privada de diferents servidors de diferents hospitals per tal de classificar diversos tipus de tumors cerebrals.

El programa dissenyat en aquest projecte intenta emular el sistema descrit anteriorment, i que, per tant, resolgui les problemàtiques de la privadesa de les dades i la distància geogràfica. Així doncs, emularà diferents servidors privats, dels quals extraurà informació d'ells a través d'agents classificadors i, mitjançant algorismes supervisats, i la mostrarà a l'usuari a través d'un agent principal.

En el següents apartats es fa menció als motius que han dut a terme la realització d'aquest projecte, als principals objectius que es pretenen assolir i a les principals tasques a realitzar. També, es fa una planificació temporal de les tasques que es volen realitzar.

1.1. Justificació del TFG.

Després d'haver realitzat totes les assignatures de l'àrea d'Intel·ligència Artificial del Grau d'Enginyeria Informàtica i haver-me endinsat en assignatures com Aprenentatge Computacional he pogut descobrir camps tant fascinants com els algorismes de predicció i els sistemes multiagent, on he volgut aprofundir els coneixements amb la realització d'aquest treball.

La primera immersió en els SMA em va permetre descobrir aquest món, en què aquest conjunt d'entitats (agents) viatgen independentment per la xarxa, intercanviant informació i realitzant tasques de forma autònoma.

A més, si a aquest sistema d'agents, hi afegim la capacitat d'aprenentatge dinàmic dels d'algorismes supervisats, provoca que el sistema es transformi en un microunivers autònom fascinant.

Finalment, considero que l'àrea d'Intel·ligència Artificial és la més encisadora, ja que, comprèn disciplines tant interessants com la programació, les matemàtiques, protocols de pas de missatges, càlculs computacionals, etc.

1.2. Objectius.

L'objectiu principal (emmarcat com objectiu didàctic) és l'aprofundiment del coneixement vers els sistemes multiagent i la realització d'un projecte de dimensions mitjanes com podria ser la tasca del dia a dia d'un enginyer informàtic.

Els objectius més específics (o tècnics) relacionats amb l'elaboració d'aquest projecte són:

- Estudi i aprenentatge extens dels sistemes multiagent.
- Aprenentatge de l'entorn de programació de SMA JADE.
- Aprofundiment en diversos algorismes de classificació.
- Estudi de l'entorn Weka per Java.
- Disseny d'una plataforma d'interacció de sistemes multiagent.
- Aprofundiment en la programació en llenguatge Java.
- Ús del disseny de programari amb mètode incremental.
- Disseny de diagrames de seqüència i UML.

1.3. Tasques.

Les tasques principals en què es compondrà la realització d'aquest treball es poden resumir en el següent esquema:

1. Planificació:
   1.1. Introducció a JADE.
   1.2. Definició projecte.
   1.3. Objectius.
   1.4. Planificació.

2. Requisits:
   2.1. Instal·lació Jade.
   2.2. Estudi manuals Jade.
   2.3. Estudi entorn Weka amb Java.
   2.4. Càlculs inferencials agents.
3. Disseny i anàlisi
   3.1. Definició casos d'ús.
   3.2. Disseny UML.
4. Implementació:
   4.1. Agents
   4.2. Comportaments.
   4.3. Resta de classes.
   4.4. *Logs* i monitoratge.
5. Proves:
   5.1. Elaboració arxius proves.
   5.2. Proves de programari.
6. Entrega Final:
   6.1. Redacció del treball.
   6.2. Elaboració de diapositives.
   6.3. Elaboració vídeo presentació.

La temporització de les diferents tasques s'estableix en el següent diagrama de Gantt:

Nombre	Fecha de inicio	Fecha de fin	Duración
1. Planificació	17/09/14	1/10/14	11
1.1. Introducció a JADE	17/09/14	24/09/14	6
1.2. Definició del projecte	25/09/14	26/09/14	2
1.3. Objectius	29/09/14	29/09/14	1
1.4. Planificació	30/09/14	1/10/14	2
2. Requisits	2/10/14	15/10/14	10
2.1. Instal·lació de JADE	2/10/14	6/10/14	3
2.2. Estudi manuals JADE	7/10/14	10/10/14	4
2.3. Estudi entorn Weka	13/10/14	14/10/14	2
2.4. Inferències	14/10/14	15/10/14	2
3. Anàlisi i Disseny	16/10/14	29/10/14	10
3.1. Definició de casos d'ús	16/10/14	21/10/14	4
3.2. Diagrama UML	22/10/14	27/10/14	4
3.3. Disseny Inferències	27/10/14	29/10/14	3
4. Codificació	30/10/14	18/11/14	14
4.1. Classes Agent	30/10/14	4/11/14	4
4.2. Comportaments	5/11/14	11/11/14	5
4.3. Resta de classes	12/11/14	14/11/14	3
4.4. Logs i monitoreig	17/11/14	18/11/14	2
5. Proves	19/11/14	25/11/14	5
5.1. Elaboració arxius proves	19/11/14	21/11/14	3
5.2. Proves de programari	24/11/14	24/11/14	1
5.3. Testeig amb Weka	24/11/14	25/11/14	2
6. Entrega Final	27/11/14	23/12/14	19
6.1. Redacció treball	27/11/14	18/12/14	16
6.2. Emmarcació producte	19/12/14	23/12/14	3

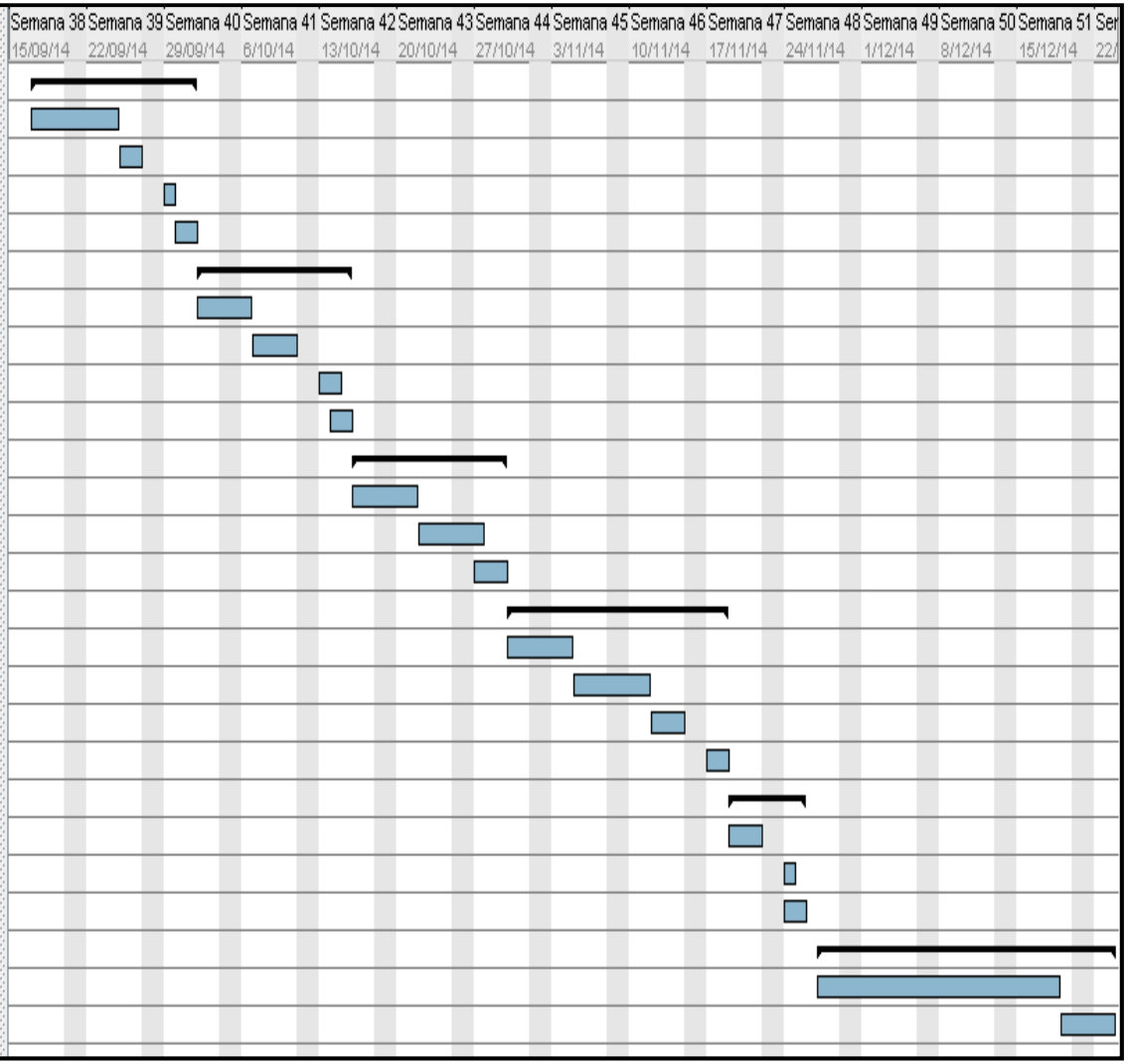

Figura 2: Diagrama de Gantt

La relació de tasques amb les diferents PACS i dates serà la següent:

PAC1 (01/10/14)	***PAC2 (29/10/14)***	***PAC3 (26/11/14)***	***Entrega Final (23/12/14)***
Punt 1	Punt 2 i 3	Punt 4 i 5	Punt 6

Tot i que s'estableix aquest pla de treball, s'estableix un marge de modificació en el contingut a lliurar en cada entrega que pot dependre de diferents factors i riscos.

1.4. Escenari.

L'escenari estarà format pels següents components:

i. Un agent manager que s'encarregarà d'interactuar amb l'usuari per crear els agents classificadors, per enviar dades als agents classificadors i per, en última instància, presentar-li els resultats finals.
ii. Diversos agents classificadors que s'entrenaran amb dades independents. Alhora, aquests agents classificadors, allotjaran un granja de algorismes supervisats amb els que extrauran la informació a les dades "privades".

El mode general de funcionament del sistema serà el següent:

i. L'usuari executarà un agent Manager que crearà diversos agents Classificadors i tants conjunts de dades independents com agents Classificadors.

ii. Els agents Classificadors, un cop creats, enviaran una notificació a l'agent Manager per comunicar-li que estan a punt per classificar dades.

iii. L'agent manager, un cop rebuda la notificació, enviarà un conjunt de dades a classificar a cada agent classificador.

iv. Els agents classificadors, per cada algorisme supervisat del que disposaran, s'entrenaran amb un conjunt independent de dades.

v. Un cop rebudes les dades a classificar del Manager, cada agent Classificador farà diverses prediccions (amb els diversos algorismes supervisats dels que disposa) que seran enviades al Manager.

vi. Finalment, l'agent Manager farà una classificació final, mitjançant un càlcul estadístic i en funció dels resultats obtinguts, que serà el que es presentarà a l'usuari.

1.5. Metodologia.

Es dissenyarà un diagrama de seqüència i un diagrama UML amb les principals classes i mètodes. Entre ambdós es tindrà l'estructura del programa.

A continuació s'establirà un conjunt de dades del qual extraure'n instàncies per a l'entrenament dels agents classificadors i del qual extreure'n instàncies per enviar a classificar. El repositori utilitzat per tal tasca serà el *UCI Machine Learning Repository.*

Un cop definit l'UML del programari s'aniran implementant les diferent classes. Aquesta implementació començarà per les classes dels diversos agents i, després i, de forma seqüencial pels comportaments de cada agent en cada escenari (o *behaviours*), després per les ontologies (segons les dades de *l'script* UCI), etc.

Un cop implementat el programa es provaran les sortides que generi.

1.6. Maquinari i programari.

El diferent maquinari i programari utilitzat serà:

- Maquinari:
    - Ordenador portàtil LENOVO B560 amb processador i3-380M, memòria RAM de 4096MB i memòria de disc HDD 320GB. El sistema operatiu emprat és Windows7 Home Premium SP1.
- Programari:
    - Diagrama de Gantt: Gantt Project.
    - Redacció: Microsoft Office 2007 (Word, Power Point).
    - Impressió: PDFCreator.
    - Entorn de programació: Eclipse Juno.
    - Còpies de seguretat: Dropbox.
    - UML: Magic Draw.

1.7. Anàlisi de riscos.

Alguns dels possibles riscos que es poden donar durant l'execució i, que poden variar les dates d'entrega del treball, poden ser:

- Avaria de l'ordinador de treball: És un risc de caràcter greu, que es pot transformar en un gran pèrdua de temps (temps necessari en arreglar l'avaria). En cas de que el període de reparació fos massa elevat, es

substituiria per un altre ordinador portàtil on el temps perdut es simplificaria a la instal·lació del programari necessari.

- Malaltia del desenvolupador (estudiant): És un risc de caràcter variable depenent de la malaltia. Pot repercutir des d'un possible retràs en la implementació del TFG segons el pla de treball (malaltia lleu), fins la no execució del TFG (malaltia greu).

1.8. Política de Seguretat.

S'estableix la política de seguretat següent:

- El treball diari s'emmagatzemarà en l'ordinador de treball i, a més, en un llapis USB.
- Cada conjunt definit de tasques, un cop realitzat, s'emmagatzemarà en un servidor remot de *Dropbox*.

2. DISSENY

Per desenvolupar el sistema descrit en apartats anteriors s'ha utilitzat el *middleware* JADE (desenvolupat en Java). Aquest, facilita el desenvolupament dels SMA sota l'estàndard FIPA, creant un o múltiples contenidors destinats als agents, on cadascun dels quals es pot executar sota un o varis sistemes.

Amb JADE, també es proporciona una implementació estàndard del llenguatge de comunicació FIPA-ACL, que facilita la comunicació entre agents i permet la detecció de serveis que es proporcionen en el sistema.

En el disseny d'aquest projecte, es crearà un agent Manager, en un contenidor, i diversos agents classificadors, en el mateix contenidor. Aquests agents classificadors, realitzaran les tasques d'extracció de dades dels seus servidors i es comunicaran amb el Manager, per passar-li la informació, mitjançant l'estàndard FIPA-ACL de JADE.

2.1. Requisits

Atès que el sistema hauria de ser distribuït en l'espai, en què cada agent classificador extreu dades d'un servidor diferent, el principal requisit que ha de complir el programari és que cada agent s'entreni amb un conjunt independent de dades (aquest conjunt de dades emularia el seu servidor).

Altres requisits funcionals són:

- Els agents han d'intercanviar informació entre ells i s'han d'entendre.
- Hi ha d'haver un algorisme que extregui les dades necessàries de cada servidor, sense violar-ne la privadesa.
- L'agent principal ha d'extreure una resultat o conclusió final a partir de total la informació rebuda.

Aquests requisits estan coberts per la plataforma JADE amb la què s'elabora el projecte. D'altres, com l'enteniment entre agents, amb l'ús d'ontologies i l'estàndard FIPA-ACL de JADE i d'altres, mitjançant els càlculs del programari dissenyat.

Alhora, l'entorn de programació JADE requereix d'un requisit de sistema com és tenir instal·lat el Java JDK7.

2.2 Diagrames

A continuació es presenten un conjunt de diagrames que representen tot el disseny lògic i que permetran una implementació més ràpida del programari.

2.2.1. Diagrama de Components

En el següent diagrama de components es poden veure les parts físiques més importants del sistema i la interacció entre elles.

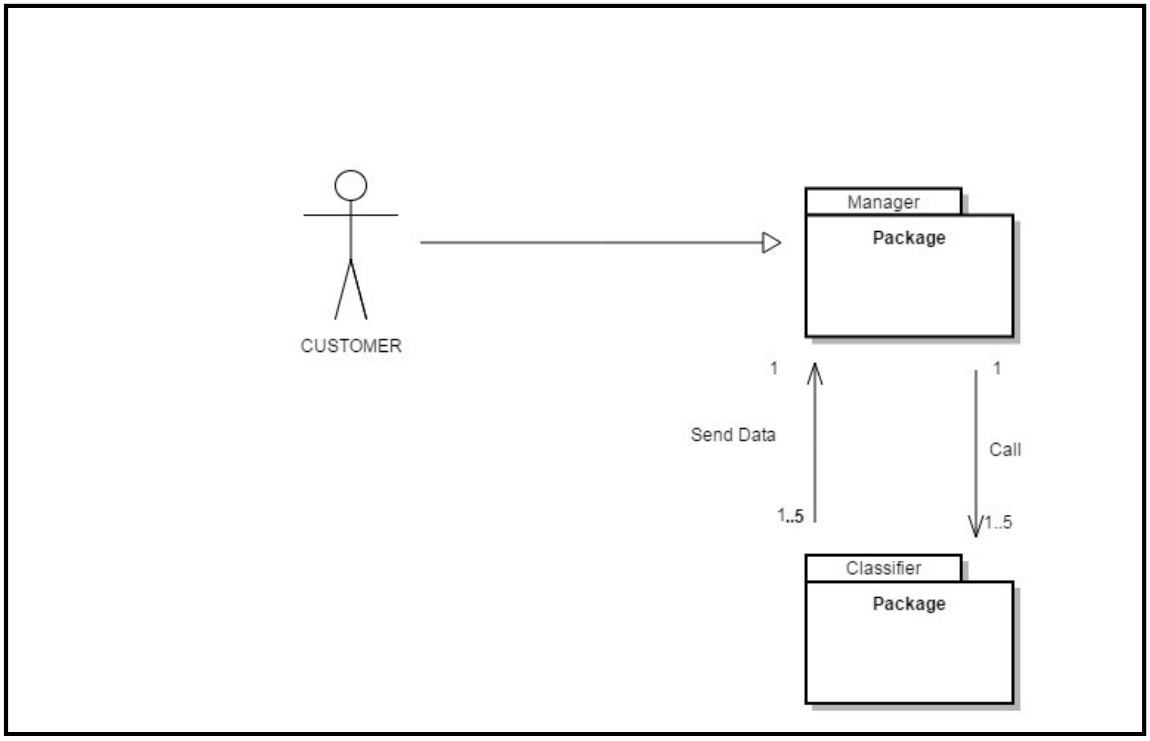

Figura 3: Diagrama de Components

2.2.2. Diagrama de Seqüència

En el següent diagrama de seqüència s'observa l'interacció temporal entre els diferents objectes segons el sistema UML implementat (més avall).

S'han representat només i, per claredat del diagrama, les classes principals. Les funcions que s'invoquen entre classes representen de forma il·lustrativa quina és l'evolució del programa.

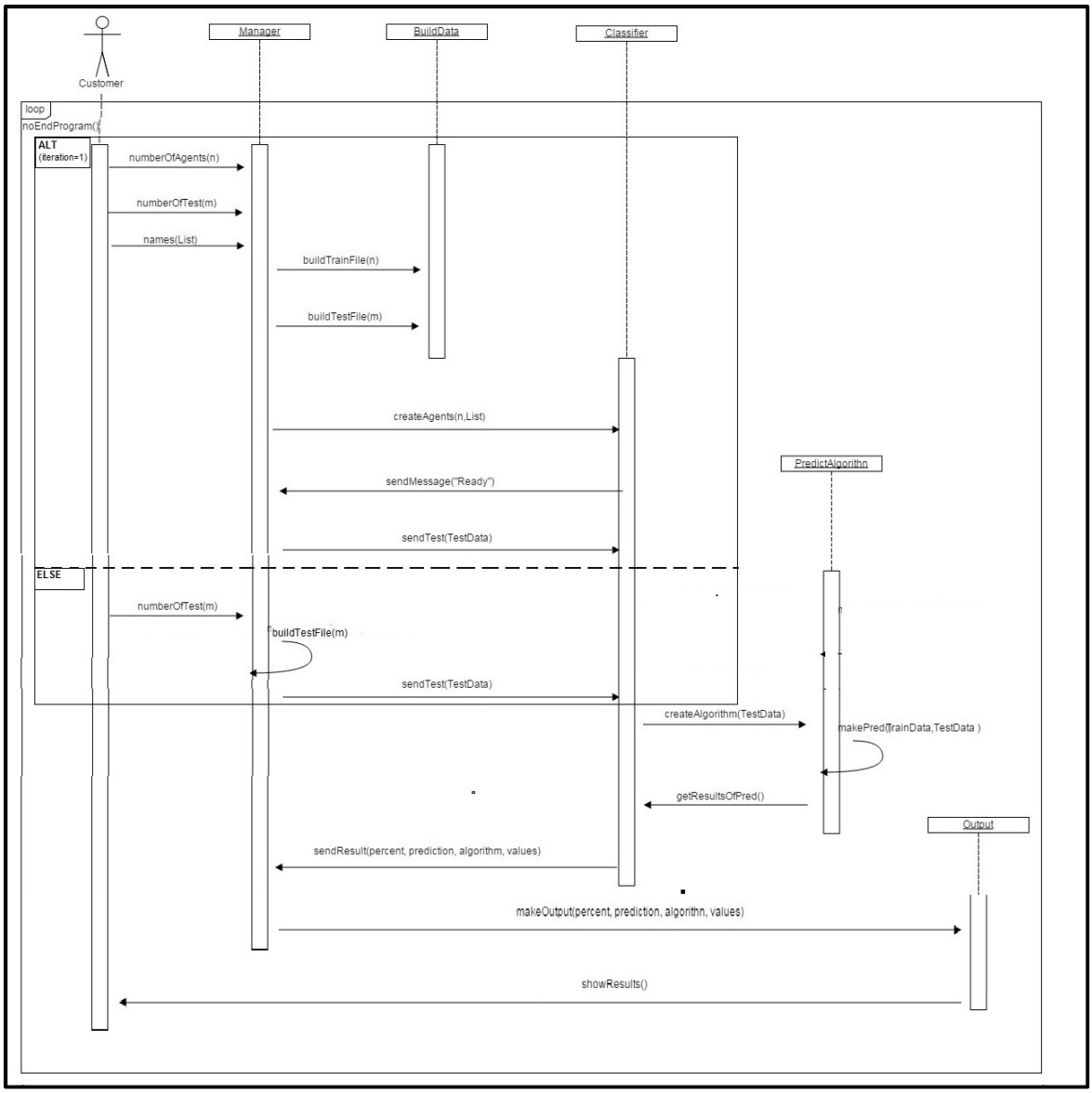

Figura 4: "Diagrama de Seqüència"

2.2.3. Diagrama de Paquets

En la següent imatge s'observa el diagrama de paquets, en què es mostren agrupades i unificades totes les relacions físiques i lògiques de totes les classes del programari, representades, cada una, amb seu *package* pertinent.

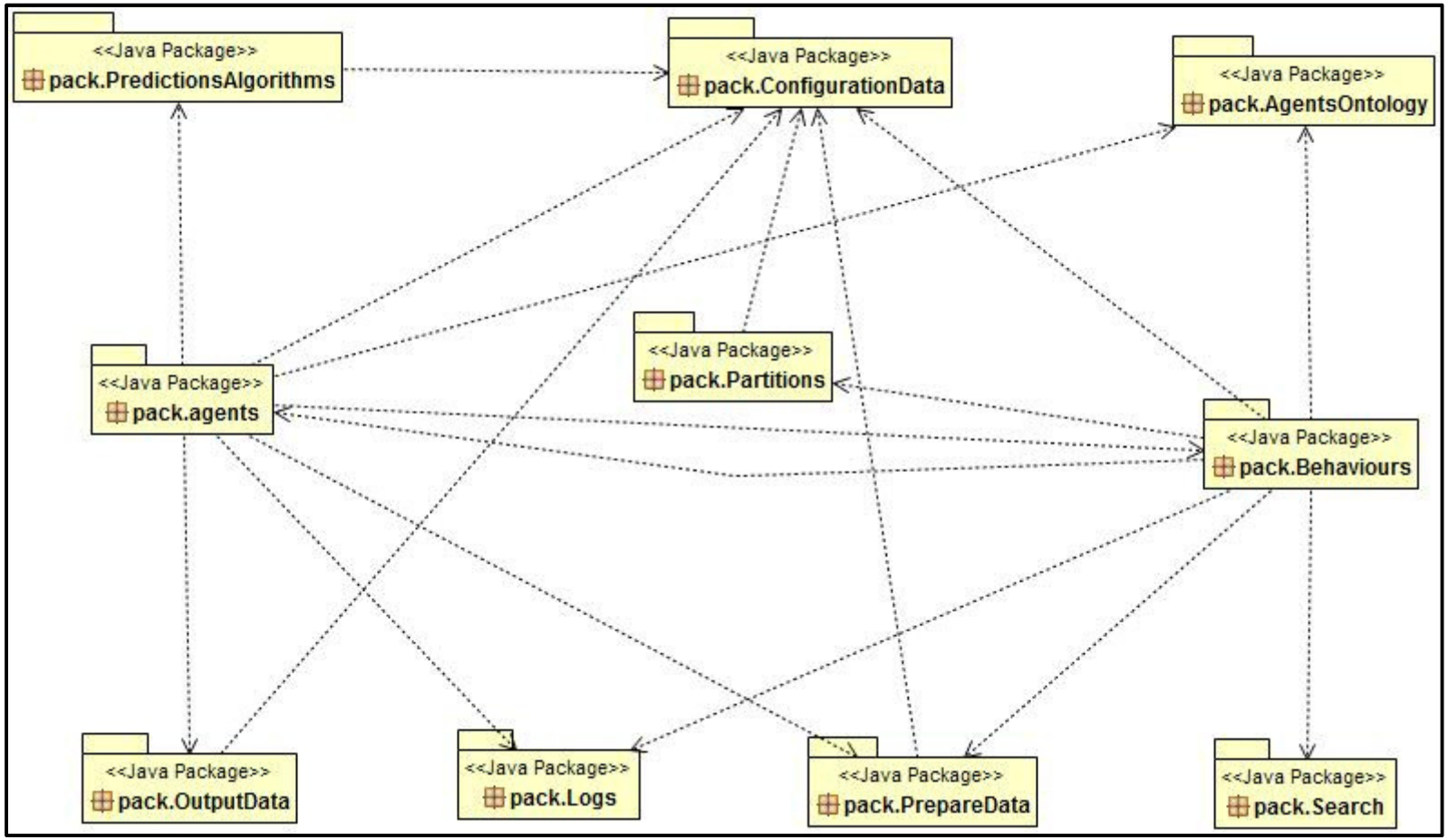

Figura 5: Diagrama de Paquets

2.2.4. Diagrama UML

En el següent diagrama UML s'observen algunes de les classes que formen el programari (les més representatives), amb els seus mètodes principals.
Si s'analitza aquest diagrama, junt amb el diagrama de seqüència, s'observa l'evolució de cada classe en una execució del programa.

**Nota: En el diagrama faltaria incloure les classes dels paquets Logs, ConfigurationData i AgentOntology. Per motius de claredat, s'ha obviat afegir aquestes classes en el diagrama. Si es vol consultar el diagrama complet es pot fer en l'Annex, punt 7.3.*

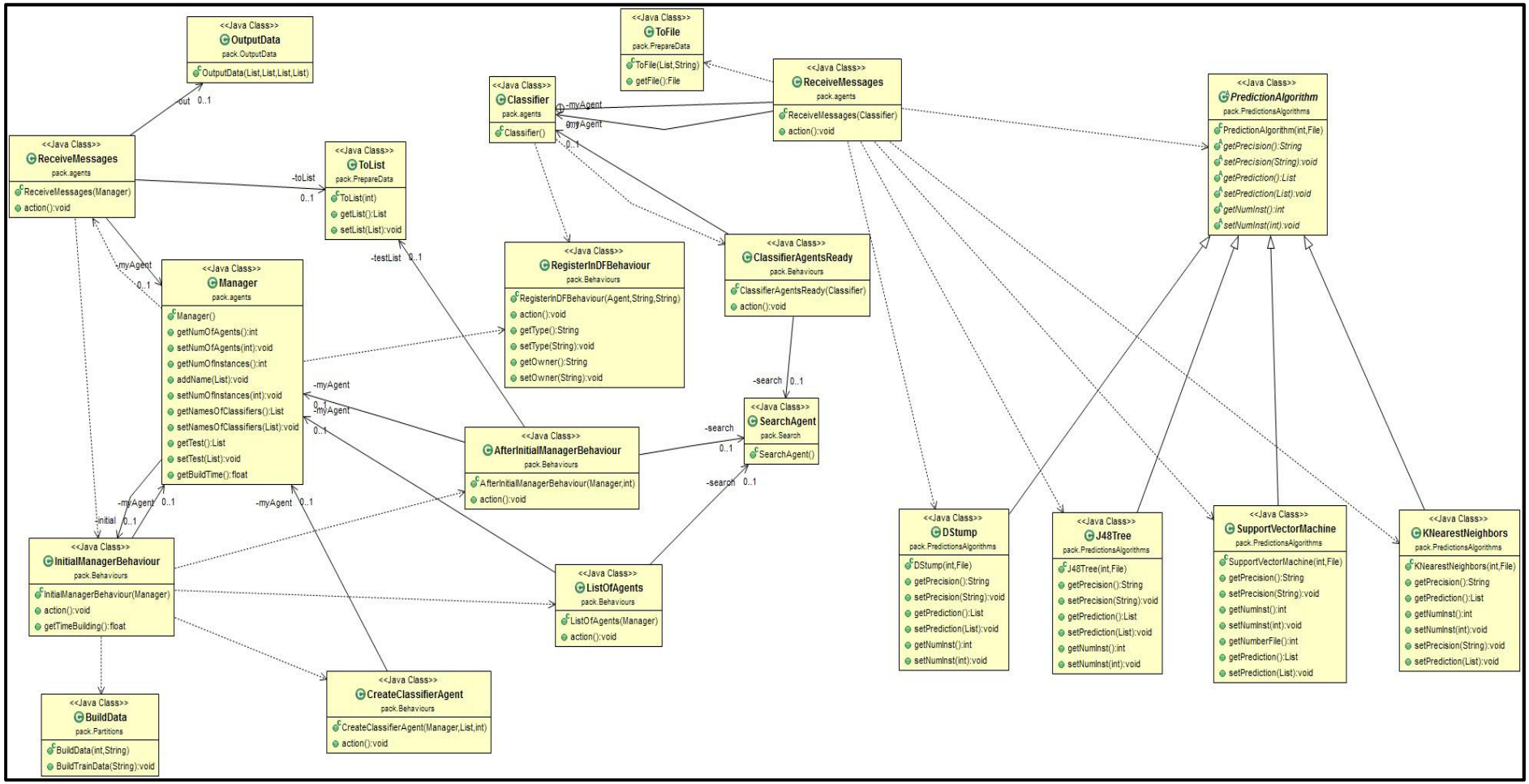

Figura 6: Diagrama UML

En resum, es pot explicar una execució del programa com:

1. S'inicia el programa.

2. S'executa el *Behaviour InitialManagerBehaviour* de Manager que demana:
    - Primer el nombre d'agents que volem crear.
    - El nombre de instàncies de les que volem saber la predicció.
    - Si volem crear un conjunt aleatori d'instàncies en el conjunts d'entrenament.
    - El nom que posarem a cada agent classificador.

3. Aquest *Behaviour* anterior crea un objecte de la classe *BuildData*. Aquest objecte crea tants fitxers train.*csv* com agents classificadors hem creat. Si hem indicat aleatorietat, crea conjunts amb diferents nombre d'instàncies. Llavors, cada agent classificador utilitzarà un d'aquests fitxers train.*csv* per entrenar-se (per tant, cada agent classificador s'entrena amb instàncies d'un conjunt independent de dades).

4. Un cop creats els agents classificadors enviaran un missatge al Manager per dir-li que estan apunt per realitzar prediccions.

5. El Manager en rebre aquests missatges "*Ready*" dels classificadors i, amb el nombre d'instàncies a classificar (del *Behaviour* inicial) crearà una instància de *ToList.* Aquesta, posarà aquest nombre d'instàncies a classificar (del fitxer principal) en un llista, que posteriorment, serà enviada a cada classificador en un nou missatge.

6. El classificadors, en rebre aquest missatge amb la llista de instàncies a classificar, crearà un arxiu test.*csv* amb un objecte de la classe *toFile* i, després, crearan els objectes que invoquen als algorismes supervisats.

7. Els algorismes supervisats amb els arxius d'entrenament i l'arxiu de la classe *toFile* faran una predicció. Un cop feta la predicció, emmagatzemaran dades com la precisió, el valor predit i el nombre de dades del conjunt d'entrenament.

8. Cada agent classificador recuperarà les dades esmentades en el punt anterior i les enviarà al Manager en un missatge.

9. El Manager, amb les dades rebudes dels classificadors, farà una crida a la classe *OutputData* que presentarà els resultats al client en un fitxer results.txt.

10. Un cop mostrats els resultats es podran repetir tantes prediccions, per aquell conjunt d'instàncies, com vulguem. Per cada nou càlcul que es vulgui fer s'hauran introduir el nombre d'instàncies que vulguem que tingui el conjunt test.csv. Si es desitja, també es podran crear mes agents classificadors amb conjunts de dades independents o mostrar els agents *online* per pantalla.

2.3. Càlcul estadístic pel resultat final.

La implementació del programari s'ha dissenyat de tal forma que si el fitxer inicial té un total d'instàncies N, al crear el conjunt d'agents classificadors 'k' es creen 'k' conjunts disjunts d'instàncies amb el mateix n/N conjunt d'instàncies cada conjunt. Per tant, cada agent tractarà amb el mateix nombre d'instàncies per elaborar les prediccions.

Si tenim en compte que el nombre d'instàncies en sistemes reals seria diferent per cada classificador, s'ha dissenyat un algorisme que crea diferents conjunts aleatòries d'instàncies per cada agent (n_i). On el

$$\sum_{i=1}^{k} \frac{n_i}{N} = N \quad k=number\ of\ agents$$

Si a més, es té en compte que cada agent classificador crida a quatre algorismes supervisats per fer les prediccions, la precisió i la classe global predita han contenir la part proporcional a cada conjunt classificador diferent.

Així doncs, per representar aquest fet i, tenint en compte el diferent nombre d'instàncies que pugui tenir cada conjunt d'entrenament de cada agent classificador, es pot calcular la precisió global final amb el següent càlcul estadístic:

$$\boldsymbol{P = \sum_{i=1}^{k} \sum_{j=1}^{4} p_{ji} \frac{n_i}{4N}}$$

On 'j' representa el nombre d'algorisme supervisat (n'hi ha quatre), la 'i' representa el nombre d'agents que van des d'un fins a 'k, la 'p_{ij}' representa la precisió unitària de cada algorisme supervisat en cada agent classificador, la 'n_i' representa el

nombre d'instàncies de cada conjunt d'entrenament i N el conjunt total d'instàncies que tenim.

La classe predita per una instància determinada pot variar segons l'algorisme utilitzat. Per tant, per determinar la classe final d'una instància a predir farem el següent còmput:

Definirem un pes per cada agent de la següent manera:

$$\boldsymbol{w_i = \frac{n_i}{4N}} \quad \forall\ i \text{ agent Classificador}$$

Llavors, definirem el valor final predit com:

$$\boldsymbol{MAX}\left(\sum_{f(i)}\sum_{g(j)}\{v_m :\ p_{ij}w_i\}\right) \quad \forall\ V_m$$

És a dir, per cada possible valor predit V_m sumarem totes aquelles precisions unitàries per cada algorisme supervisat i per cada agent que tinguin el mateix valor. De tots els possibles valors obtinguts després de fer la suma, es retornarà aquell valor que tingui màxima suma.

Les funcions f(i) i g(j) són unes funcions que retornen els 'i' (agents) i els 'j' (algorismes), repectivament, tals que el valor predit V_m sigui el mateix.

3. Implementació

En aquest apartat s'explica tota la implementació seguida segons el diagrama de paquets.

Agents:

El primer paquet implementat és el paquet d'agents (*pack.agents*). En aquest paquet trobem dues classes principals. La classe *Manager* i la classe *Classifier*.

Tant la classe *Manager* com la classe *Classifier*, segons l'entorn JADE, han de cridar el mètode *setup*() en el qual s'afegiran, entre d'altres paràmetres, els comportaments que definiran l'agent. Totes dues classes han d'heretar de la classe *Agent* i en la crida al mètode *setup*() han de registrar un llenguatge i una ontologia per la comunicació.

A continuació es veu la implementació d'aquest mètode per la classe *Manager* (la classe *Classifier* és similar, només amb la variació dels *behaviours* creats):

```
public class Manager extends Agent  {

      private SLCodec codec=new SLCodec();
      private Ontology onto=OntoOntology.getInstance();
      Configuration conf=new Configuration();

    protected void setup(){

            getContentManager().registerLanguage(codec);
             getContentManager().registerOntology(onto);

             SequentialBehaviour sb = new SequentialBehaviour();
             RegisterInDFBehaviour rDF=new RegisterInDFBehaviour(this,"Manager",
this.getName());
             sb.addSubBehaviour(rDF);
             addBehaviour(sb);
             addBehaviour(new InitialManagerBehaviour(this););
             addBehaviour(new ReceiveMessages(this));
   }
}
```

Figura 7: Mètode setup() de l'agent Manager

Behaviours:

Alguns dels comportaments que podem trobar en *JADE* són:

- Comportaments *OneShot*: comportaments que s'executen de manera instantània i una sola vegada.
- Comportaments Cíclics: són aquells que mai es treuen del comportament de l'agent i els quals amb el seu mètode *action*() sempre s'executen el mateix codi sense que mai acabin.
- Comportaments genèrics: són una mica més complexes que la resta. El codi que executen depèn de *l'status* de l'agent y eventualment finalitzen la seva execució.

Si en el programari dissenyat analitzem el paquet *pack.Behaviours* trobem els diferents comportaments que implementen els agents. Així doncs, en aquest paquet trobem comportaments per als dos tipus d'agents (*Manager* i *Classifier*).

Els diferents *behaviours* que podem trobar són:

- *InitialManagerBehaviour (per Manager)* que hereta del comportament *OneShot:* és el primer comportament del manager al executar-se el programa. Des d'aquest, l'usuari té el control del programari.
  Principalment té dues opcions:
  - El executar-lo en primera instància, ens permet crear 'n' agents i un test per predir amb 'm' instàncies, llistar els classificadors del sistema (que en executar la primera vegada són 0) o acabar el programa (deregistrant el Manager del servei DF).
    A més, al crear els 'n' agents, crea 'n' particions en un fitxer que emularan els conjunts independents de dades (o servidors).
  - En posteriors execucions, permet crear agents classificadors d'un en un, llistar els classificadors *online* del sistema o enviar nous *test*s a classificar.

- *AfterInitialManagerBehaviour* (per *Manager*) que hereta del comportament *OneShot*: aquest comportament es dóna per execucions posteriors a la primera, on els agents classificadors ja estan creats. Principalment, busquen els classificadors *online* del sistema i els hi envia un conjunt test per classificar.

- *ListOfAgents* (per *Manager*) que hereta de *OneShot*: aquest comportament llista tots els classificadors creats fins aquell moment.

- *CreateClassifierAgent* (per *Manager*) que hereta de *OneShot*: comportament que crea diferents agents classificadors.

- *ClassifierAgentReady* (per *Classifier*) que hereta de *OneShot*: és un comportament que, un cop creat l'agent classificador, envia un missatge a l'agent *Manager* per indicar-li alguns paràmetres (com el nombre d'agents dels sistema) i, principalment, avisar-lo de que ha estat creat i està apunt per fer classificacions.

- *RegisterInDFBehaviour* (per *Manager* i *Classifier*) que hereta de *OneShot*: aquest comportament registra en el DF els agents, un cop creats.

- *ReceiveMessage (per Manager i Classifier)* que hereta *de CyclicBehaviour:* aquest comportament, de forma diferent per cada tipus d'agent, provoca les diferents accions que faran el Manager i el Classificador a mesura rebin missatges l'un de l'altre.

Partitions:

En aquest paquet (*pack.partitions*) hi ha una classe *BuildData*, que s'encarrega de crear diferents conjunts independents de dades, tants com agents.

Quan s'executa per primera vegada el programa, aquest demana el nombre d'agents i si es vol nombre aleatoris d'instàncies en cada conjunt. La classe *BuildData* construeix tants conjunts independents de dades seguint algunes condicions:

- Si es demana aleatorietat en el nombre d'instàncies, aquest classe crea els *train* sets amb un nombre aleatori entre 100 i el màxim nombre d'instàncies del fitxer pare (1723).
  El 100 s'ha determinat així perquè, posteriorment, els algorismes classificadors, per entrenar-se amb els *train* creats, requeriran un nombre mínim d'instàncies.
- Si no es demana aleatorietat, el nombre d'instàncies que tindrà cada arxiu *train* serà el mateix que nombre d'instàncies de l'arxiu pare dividit pel nombre d'agents.

Per tal d'evitar posteriors errors en l'entrenament dels algorismes supervisats, a cada arxiu *train*, se li afegeixen també, quatre instàncies que contenent tots els tipus de valors que poden prendre els atributs de l'arxiu pare.

PredictionsAlgorithms:

En aquest paquet (*pack.PredictionsAlgorithms*) hi ha tot el conjunt d'algorismes supervisats que fan les prediccions dels tests enviats pel *Manager*. Cada algorisme realitza la seva predicció particular amb el conjunt de dades del seu servidor (el seu *train.csv* pertinent) i el conjunt de dades test (a avaluar) que li envia el *Manager*.

En aquest paquet trobem cinc classes; les quatre primeres hereten de l'última:

- *DStump*: aquest classe crea un objecte *DecisionStump* amb el que farà un predicció d'un conjunt test, previ entrenament. Per crear aquest objecte s'ha d'importar les llibreries Weka que contenen aquest algorisme.
- *J48Tree*: aquest classe, igual que l'anterior, crea un objecte *J48* amb el que farà un predicció d'un conjunt test, previ entrenament. També requereix de l'importació de les llibreries Weka per Java.
- *KNearestNeighbors*: aquest classe, igual que les anteriors, crea un objecte *KNN* amb el que farà un predicció d'un conjunt test, previ entrenament. També requereix de l'importació de les llibreries Weka per Java
- *SupporVectorMachine*: aquest classe, igual que la resta d'aquest paquet, crea un objecte *SMO* amb el que farà un predicció d'un conjunt test, previ entrenament. També requereix de l'importació de les llibreries Weka per Java.
- *PredictionAlgorithm*: és una classe abstracta que conté els principals mètodes de les quatre anteriors. És la classe cridada per l'agent Classificador per realitzar les prediccions.

En la següent figura mostra com la classe *Classifier* calcula les prediccions dels seus servidors a través d'aquest quatre algorismes supervisats que té allotjats.

Ho fa am la creació d'un objecte de cada classe de cada algorisme supervisat al que li indica per paràmetres el test que vol predir i el seu nombre de servidor (train.csv).L'instància es crea invocant un classe que hereta de *PredictionAlgorithm*.

```
class ReceiveMessages extends CyclicBehaviour {

       //more code

        …

   public void action() {
```

```
        //more code

       //Computes predictions with Decision Stump
       PredictionAlgorithm ds=new DStump(numberOfAgent,file);
       String prec=ds.getPrecision();
       precision.add(prec);
       trainNumber=ds.getNumInst();

       //Computes predictions with  J48
        PredictionAlgorithm j48=new J48Tree(numberOfAgent,file);
        prec=j48.getPrecision();
         precision.add(prec);

        //Computes predictions with KNN
       PredictionAlgorithm knn=new KNearestNeighbors(numberOfAgent, file);
       prec=knn.getPrecision();
       precision.add(prec);

      // Computes predictions with SMO
      PredictionAlgorithm smo=new SupportVectorMachine(numberOfAgent, file);
       prec=smo.getPrecision();
       precision.add(prec);

       // more code
       ...
   }}
```

Figura 8: Instanciació dels algorismes supervisats.

PrepareData:

Aquest paquet (*pack.PrepareData*) està dissenyat per resoldre la incompatibilitat entre l'enviament de missatges del *Manager* i la classificació del *test set* dels algorismes supervisats.

Les instàncies classificades pels algorismes supervisats en els Classificadors i les instàncies enviades pel *Manager* són les mateixes, però els algorismes supervisats necessiten un fitxer llegible (en aquest cas un *.csv*) i, el *Manager* ha d'enviar les dades en un format acceptable per l'ontologia (en aquest cas una llista *d'Strings*).

Trobem dues classes que són les següents:

- ToFile: aquesta classe es crida en la classe *ReceiveMessages* de *Classifier* i, principalment, s'encarrega de convertir la llista d'instàncies rebudes del *Manager* en un fitxer *test.csv*. Aquest fitxer, a posteriori, podrà ser llegit per l'algorisme supervisat.

- ToList: aquesta classe crea una llista aleatòria amb el nombre d'instàncies introduïdes en la primera execució del programa, a partir del fitxer pare (*car.csv).*

Search:

En aquest paquet (*pack.Search*) trobem una única classe. Aquesta classe, *SearchAgent* és l'encarregada de buscar els agents que hi ha en el *DFService* i que compleixen unes certes condicions. En aquesta cerca, les condicions de cerca són el tipus d'agent que són ("*Manager*" o "*Classifier*") i el tipus.

Aquesta classe, funciona, previ registre del servei dels agents en el *DFService*, que en aquest cas s'ha realitzat en les primeres accions (*behaviours*) dels agents un cop creats.

ConfigurationData:

En aquest package (pack.ConfigurationData) hi ha una única classe, *Configuration* que llegeix tots els paràmetres i configuracions, que l'usuari pot modificar abans de l'execució.

Podem trobar paràmetres com les rutes als arxius *train* i *test*, a l'arxiu pare (car.csv) i a l'arxiu *results.txt*. També, podem trobar les configuracions del *log* (si volem o no registrar tots els esdeveniments que succeeixin durant l'execució) i la verbositat (que mostra per pantalla tots els esdeveniments que succeeixen).

Amb una instància d'aquesta classe, i els seus *getters* (que retornen les dades esmentades anteriorment), podem obtenir la dada desitjada on vulguem.

OutputData:

En aquest paquet (*pack.OutputData*) trobem una classe, *OutputData* que, amb totes les dades rebudes dels agents classificadors fa els càlculs pertinents per determinar el millor valor d'entre tots els rebuts, així com, els càlculs de precisions unitàries de cada millor valor predit i la precisió global.

Es crida des del *CyclicBehaviour ReceiveMessages* en el *Manager*. Un cop el Manager ha rebut els resultats de tots el missatges amb les dades dels agents classificadors, els utilitza per crear una instància de la classe. Aquesta instància, genera els resultats finals i els escriu en el fitxer de sortida *results.txt.*

Logs*:*

Aquest paquet (*pack.Log*) s'encarrega de generar un arxiu log.txt en el què hi ha escrit tots els events que es donen en l'execució del programa.

Aquesta classe llegeix la configuració desitjada d'un arxiu *log4j.properties*. Algunes d'aquestes propietats són la ruta on es crearà l'arxiu *log.txt* i la seva mida en Kb.

De l'arxiu *configuration.prop*erties llegeix si es vol *loggejar* l'execució o executar en mode *verbose*.

En la següent imatge es pot veure una sortida generada en l'arxiu *log.txt*:

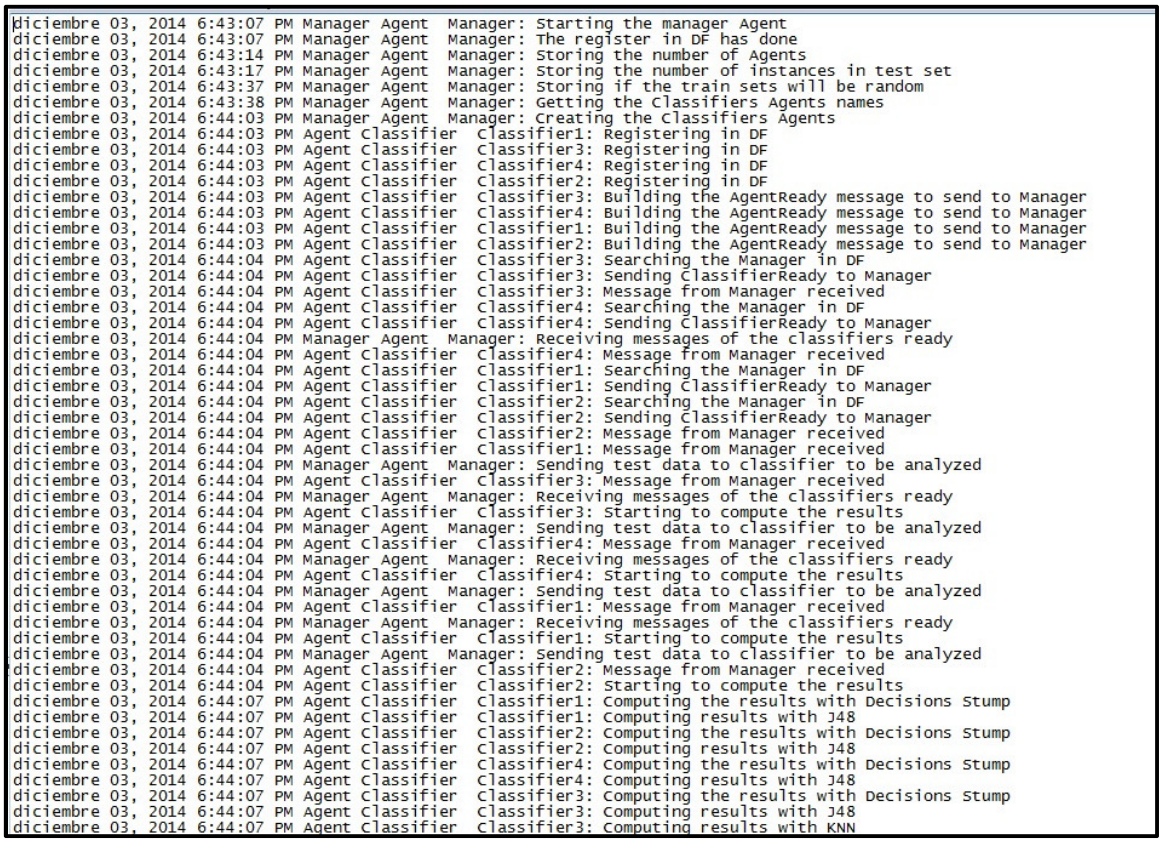

Figura 9: Arxiu log.txt

AgentsOntology:

Aquest paquet (pack.AgentsOntology) conté quatre classes. Està pensat, fonamentalment perquè els agents es puguin entendre entre ells. Així doncs, els agents quan s'envien dades, ho fan amb l'ús d'una ontologia.

Quan es crea un agent se li ha de registrar una ontologia i un llenguatge per a la posterior comunicació. Per al projecte, aquestes accions s'ha implementat en el mètode setup() de cada agent, tal i com mostra la següent imatge:

```
public class Classifier extends Agent{

	private Codec codec=new SLCodec();
	private Ontology onto =OntoOntology.getInstance();
	// more code
	...
	protected void setup(){

		getContentManager().registerLanguage(codec);
		getContentManager().registerOntology(onto);
		// more code
		...
	    }
```

Figura 10: Registre d'ontologia i llenguatge en Manager

També, cada cop que s'envia un missatge, se li ha de registrar una ontologia i un llenguatge, tal i com mostra la següent imatge:

```
//building a message to send
ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
msg.setLanguage(codec.getName());
msg.setOntology(onto.getName());

//  more code
   ...
msg.setSender(myAgent.getAID());
myAgent.send(msg);
```

Figura 11: Registre d'ontologia i llenguatge a un missatge

Uns exemple de missatge per als resultats predits que envia un Classificador al Manager (per un conjunt test de sis instàncies) és:

```
((action (agent-identifier :name Manager@192.168.0.101:1099/JADE :addresses
(sequence http://Oscar-PC:7778/acc)) (ClassificationsData :KNNPredicted
(sequence unacc unacc unacc unacc unacc unacc) :percentatge (sequence "100.0"
"100.0" "100.0" "100.0") :DsPredicted (sequence unacc unacc unacc unacc unacc
unacc) :SMOPredicted (sequence unacc unacc unacc unacc unacc unacc) :trainNumber
547 :J48Predicted (sequence unacc unacc unacc unacc unacc unacc))))
```

Figura 12: Missatge amb resultats de Classificador al Manager

En aquest missatge es veu la forma en què s'envien les dades. Es veuen diferents vectors que emmagatzemen els valors predits per cada algorisme supervisat instanciats en un agent classificador, un vector amb les precisions de cada algorisme en aquella avaluació i, finalment, un enter amb el nombre d'instàncies del conjunt *train*.

Un exemple de missatge que envia el classificador quan s'ha creat, per informar al Manager de que està a punt per classificar, és el següent:

```
((action (agent-identifier :name Manager@192.168.0.101:1099/JADE :addresses
(sequence http://Oscar-PC:7778/acc)) (AgentReady :InstancesOfTest 6
:InstancesOfAgents 2)))
```

Figura 13: Missatge AgentReady d'un Classificador al Manager

Finalment, un exemple de missatge que envia el Manager als classificador amb els conjunt test (de sis instàncies) per a ser predit és:

```
((action (agent-identifier :name Classifier1016@192.168.0.101:1099/JADE
:addresses (sequence http://Oscar-PC:7778/acc)) (ElementsToClassify :TestList
(sequence high,high,5more,2,med,high,unacc low,low,4,2,big,low,unacc
low,med,2,more,small,high,unacc high,low,4,2,small,high,unacc
med,high,5more,4,big,low,unacc vhigh,med,4,2,big,med,unacc))))
```

Figura 14: Missatge ElementsToClassify de Manager a Classificador

Per crear les classes *java* de l'ontologia s'han utilitzat les eines *Protégé* i *BeanGenerator*. L'ús d'aquestes dues eines, ens permet crear les classes *java* a partir de l'ontologia dissenyada amb l'eina *Protégé*.

El problema d'aquest mètode per generar les classes *java* de l'ontologia és que ens limita a l'ús dels tipus d'objectes que té el *Protégé* per crear l'ontologia. Per tant, no es poden enviar objectes del tipus File com podrien ser l'arxiu *test.csv*. sinó que s'ha reestructurar l'informació a tipus d'objectes *String*, enters, etc. o vectors d'aquests.

4. Proves

El programari s'ha dissenyat per a classificar vehicles. Cada instància de l'arxiu font que consulta el programari, té sis atributs nominals, que poden prendre els valors entre parèntesi:

- preu (*vhigh, high, med, low*).
- manteniment (*vhigh, high, med, low*).
- portes (*2, 3, 4 , 5more*).
- persones (*2, 4, 5more*).
- capacitat del portaequipatges (*small, med, big*).
- seguretat (*low, med, high*).

La classe final de cada instància, determina el tipus d'accessibilitat que té el vehicle (*unacc, acc, good, vgood*). Aquesta, també és la classe que es voldrà predir en el conjunts test que s'enviaran als classificadors.

Per realitzar les proves de programari, s'han creat diferents configuracions de programari, en les què hi ha diferents nombres d'agents classificadors, entrenats amb diferents conjunts d'instàncies i, als que s'han enviat diferents conjunts de dades per classificar.

Aquestes configuracions del sistema per les proves són: proves amb 1, 2, 3, 4 i 5 agents diferents que s'entrenen amb conjunt diferents de dades i diferents conjunts d'instàncies *test* per classificar. En aquestes proves, els conjunts *test* han pres valors d'1,3,5 i 8 instàncies.

A continuació, es mostra les característiques de les sortides (en forma de taula) que s'han generat amb el programari:

Els valors finals, en les prediccions de les taules de predicció dels algorismes supervisats, estan abreujats per:

1. unaac → 'u'
2. ucc → 'a'
3. good → 'g'
4. vgood → 'v'

La llegenda de les taules que es presenten per cada prova és:

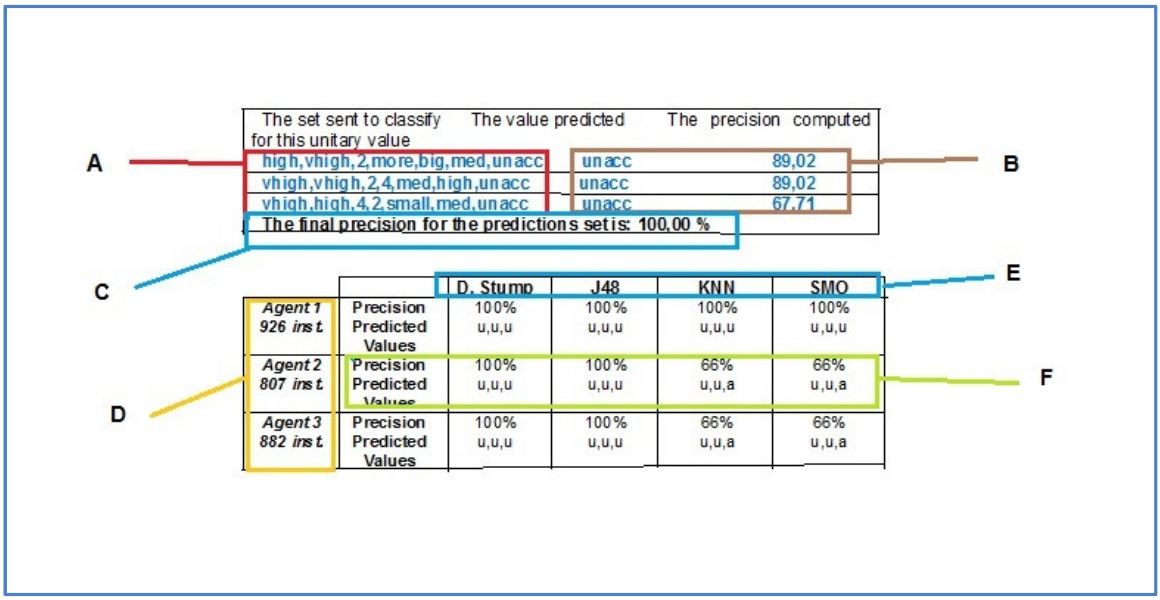

Figura 15: Llegenda dels quadres de proves del programari

On cada quadre de color representa el següent:

- **A**: És el conjunt test que s'ha enviat als agents classificadors per ésser classificat. Cada línia representa cada instància del conjunt test.
- **B**: Per cada línia es presenta el valor de classe final que el programari ha predit, junt amb la precisió unitària associada a aquell valor (calculada segons la fórmula de l'apartat 2.3)
- **C**: Mostra la precisió global del sistema. O també, els valors predits correctament dividit pel total de valors predits.
- **D**: Mostra el total d'agents classificadors que hi havia al sistema i que han generat les sortides del quadre B. Per cada agent, es mostra el total d'instàncies amb les què s'ha entrenat.
- **E**: Mostra els quatre algorismes supervisats que cada agent classificador té allotjats i que fan les prediccions.
- **F**: Aquests quadres, mostren, les precisions i els valors predits per l'agent. És a dir, per aquest quadre en concret:
    i. L'algorisme supervisat *Decision Stump* de l'agent 2, que ha fet servir 807 instàncies pel seu entrenament ha generat, pel conjunt d'instàncies del quadre A, una predicció de valors (*u,u,u*) o que és el mateix (*unacc, unacc, unacc*). La precisió de la predicció és 100%.
    ii. L'algorisme supervisat *J48* de l'agent 2, que ha fet servir 807 instàncies pel seu entrenament ha generat, pel conjunt d'instàncies del quadre A, una predicció de valors (*u,u,u*) o que és el mateix (*unacc, unacc, unacc*). La precisió de la predicció és 100%.

iii. L'algorisme supervisat *KNN* de l'agent 2, que ha fet servir 807 instàncies pel seu entrenament ha generat, pel conjunt d'instàncies del quadre A, una predicció de valors (*u,u,a*) o que és el mateix (*unacc, unacc, acc*). La precisió de la predicció és 66%.

iv. L'algorisme supervisat *SMO* de l'agent 2, que ha fet servir 807 instàncies pel seu entrenament ha generat, pel conjunt d'instàncies del quadre A, una predicció de valors (*u,u,u*) o que és el mateix (*unacc, unacc, acc*). La precisió de la predicció és 66%.

4.1. Output per 1 Agent Classificador

Per tot l'apartat 4.1 següent, el sistema conté un agent classificador. El nombre d'instàncies enviades a classificar en test varia tal i com es mostra a continuació:

Una instància enviada a classificar:

The set sent to classify for this unitary	The value predicted	The precision computed
value low,high,2,2,med,med,unacc	**unacc**	**100,00**
The final precision for the predictions set is: 100,00 %		

(1732 inst.)	**Decision Stump**	**J48**	**KNN**	**SMO**
Precision	100%	100%	100%	100%
Predicted Values	u	u	u	u

Tres instàncies enviada a classificar:

The set sent to classify	The value predicted	The precision computed for this unitary value
high,low,5more,2,med,high,unacc	**unacc**	**91,50**
high,vhigh,2,2,small,low,unacc	**unacc**	**91,50**
vhigh,med,4,4,big,low,unacc	**unacc**	**75,00**
The final precision for the predictions set is: 100,00 %		

(1732 inst.)	Decision Stump	J48	KNN	SMO
Precision	100%	100%	66.6%	100%
Predicted Values	u,u,u	u,u,u	u,u,a	u,u,u,

Cinc instàncies enviada a classificar:

The set sent to classify	The value predicted	The precision computed for this unitary value
low,high,3,more,small,low,unacc	**unacc**	**65,00**
high,low,3,2,med,high,unacc	**unacc**	**65,00**
low,med,5more,2,med,high,unacc	**unacc**	**50,00**
high,high,2,4,small,high,acc	**unacc**	**65,00**
high,low,2,2,small,med,unacc	**unacc**	**35,00**
The final precision for the predictions set is: 80,00 %		

(1732 inst.)	Decision Stump	J48	KNN	SMO
Precision	80%	60%	60%	60%
Predicted Values	u,u,u,u,u	u,u,u,u,a	u,u,u,u,a	u,u,a,u,u

Vuit instàncies enviades a classificar:

The set sent to classify:	The value predicted:	The precision computed for this unitary value:
high,high,4,2,big,high,unacc	**unacc**	**84,38**
high,high,2,2,big,high,unacc	**unacc**	**84,38**
high,low,4,2,med,high,unacc	**unacc**	**84,38**
low,high,4,2,small,low,unacc	**unacc**	**84,38**
med,low,2,4,small,low,unacc	**unacc**	**65,63**
high,high,4,2,med,low,unacc	**unacc**	**84,38**
low,high,5more,more,med,high,vgood	**unacc**	**84,38**
vhigh,high,3,4,med,med,unacc	**unacc**	**84,38**
The final precision for the predictions set is: 87,50 %		

(1732 inst.)	Decision Stump	J48	KNN	SMO
Precision	87.5%	87.5%	87.5%	75%
Predicted Values	u,u,u,u,u,u,u,u	u,u,u,u,u,u,u,u	u,u,u,u,u,u,u,u	u,u,u,u,v,u,u,u

4.2 Output per 2 Agents Classificadors

Per tot l'apartat 4.2 següent, el sistema conté dos agents classificadors. El nombre d'instàncies enviades a classificar en test varia tal i com es mostra a continuació:

Una instància enviada a classificar:

The set send to classify this unitary value	The value predited	The precision computed for
vhigh,low,3,2,small,high,unacc	**unacc**	**100,00**
The final precision for the predictions set is: 100,00 %		

		D. Stump	J48	KNN	SMO
Agent 1 900 inst.	**Precision**	100%	100%	100%	100%
	Predicted Values	u	u	u	u
Agent 2 1306 inst.	**Precision**	100%	100%	100%	100%
	Predicted Values	u	u	u	u

Tres instàncies enviades a classificar:

The set sent to classify this unitary value	The value predicted	The precision computed for
low,high,3,4,med,low,unacc	**unacc**	**66,00**
low,vhigh,3,4,small,high,acc	**unacc**	**66,00**
low,med,4,4,med,low,unacc	**unacc**	**66,00**
The final precision for the predictions set is: 66,67 %		

		D. Stump	J48	KNN	SMO
Agent 1 900 inst.	**Precision**	66.6%	66.6%	66.6%	66.6%
	Predicted Values	u,u,u	u,u,u	u,u,u	u,u,u
Agent 2 1306 inst.	**Precision**	66.6%	66.6%	66.6%	66.6%
	Predicted Values	u,u,u	u,u,u	u,u,u	u,u,u

Cinc instàncies enviades a classificar:

The set sent to classify	The value predicted	The precision computed for this unitary value
med,high,3,2,big,med,unacc	**unacc**	**92,04**
low,low,2,2,small,low,unacc	**unacc**	**92,04**
med,high,2,more,small,med,unacc	**unacc**	**92,04**
low,med,4,2,med,med,unacc	**unacc**	**92,04**
vhigh,vhigh,2,4,big,med,unacc	**unacc**	**60,20**
The final precision for the predictions set is: 100,00 %		

		D. Stump	J48	KNN	SMO
Agent 1 900 inst.	**Precision**	100%	100%	100%	80%
	Predicted Values	u,u,u,u,u	u,u,u,u,u	u,u,u,u,u	u,u,u,u,a
Agent 2 1306 inst.	**Precision**	100%	100%	80%	80%
	Predicted Values	u,u,u,u,u	u,u,u,u,u	u,u,u,u,a	u,u,u,u,g

Vuit instàncies per classificar:

The set sent to classify:	The value predicted:	The precision computed for this unitary value:
vhigh,high,2,2,med,high,unacc	**unacc**	**62,42**
high,low,4,4,big,med,acc	**unacc**	**36,57**
high,high,3,4,small,med,unacc	**unacc**	**62,42**
vhigh,med,5more,more,big,med,acc	**acc**	**49,92**
med,med,5more,2,small,high,unacc	**unacc**	**54,81**
low,vhigh,3,2,med,low,unacc	**unacc**	**62,42**

high,med,5more,4,med,high,acc	unacc	62,42
low,med,3,4,big,high,vgood	acc	33,87
The final precision for the predictions set is: 62,50 %		

		D. Stump	J48	KNN	SMO
Agent 1 900 inst.	**Precision**	50%	75%	75%	62.5%
	Predicted Values	u,u,u,u,u,u,u,u	u,a,u,a,u,u,u,a	u,a,u,a,u,u,u,a	u,a,u,a,a,u,u,a
Agent 2 1306 inst.	**Precision**	50%	62.5%	62.5%	62.5%
	Predicted Values	u,u,u,u,u,u,u,u	u,u,u,a,u,u,u,u	u,u,u,a,u,u,u,u	u,u,u,a,u,u,u,u

4.3. Output per 3 Agents Classificadors

Per tot l'apartat 4.3 següent, el sistema conté tres agents classificadors. El nombre d'instàncies enviades a classificar en test varia tal i com es mostra a continuació:

Una instància enviada a classificar:

The set sent to classify	The value predicted	The precision computed for this unitary value
high,vhigh,5more,4,small,high,unacc	unacc	100,00
The final precision for the predictions set is: 100,00 %		

		D. Stump	J48	KNN	SMO
Agent 1 926 inst.	**Precision**	100%	100%	100%	100%
	Predicted Values	u	u	u	u
Agent 2 807 inst.	**Precision**	100%	100%	100%	100%
	Predicted Values	u	u	u	u
Agent 3 882 inst.	**Precision**	100%	100%	100%	100%
	Predicted Values	u	u	u	u

Tres instàncies per classificar:

The set sent to classify for this unitary value	The value predicted	The precision computed
high,vhigh,2,more,big,med,unacc	**unacc**	**89,02**
vhigh,vhigh,2,4,med,high,unacc	**unacc**	**89,02**
vhigh,high,4,2,small,med,unacc	**unacc**	**67,71**
The final precision for the predictions set is: 100,00 %		

		D. Stump	J48	KNN	SMO
Agent 1 926 inst.	**Precision**	100%	100%	100%	100%
	Predicted Values	u,u,u	u,u,u	u,u,u	u,u,u
Agent 2 807 inst.	**Precision**	100%	100%	66%	66%
	Predicted Values	u,u,u	u,u,u	u,u,a	u,u,a
Agent 3 882 inst.	**Precision**	100%	100%	66%	66%
	Predicted Values	u,u,u	u,u,u	u,u,a	u,u,a

Cinc instàncies enviades a classificar:

The set sent to classify for this unitary value	The value predicted	The precision computed
high,low,3,4,med,low,unacc	**unacc**	**66,74**
high,med,2,4,small,med,unacc	**unacc**	**66,74**
med,low,2,more,med,med,acc	**unacc**	**66,74**
med,med,5more,2,big,low,unacc	**unacc**	**40,23**
med,high,5more,more,big,low,unacc	**unacc**	**40,23**
The final precision for the predictions set is: 80,00 %		

		D. Stump	J48	KNN	SMO
Agent 1 926 inst.	**Precision**	80%	80%	40%	40%
	Predicted Values	u,u,u,u,u	u,u,u,a,a	u,u,u,a,a	u,u,u,a,a
Agent 2 807 inst.	**Precision**	80%	80%	40%	40%
	Predicted Values	u,u,u,u,u	u,u,u,a,a	u,u,u,a,a	u,u,u,a,a
Agent 3 882 inst.	**Precision**	80%	80%	80%	80%
	Predicted Values	u,u,u,u,u	u,u,u,u,u	u,u,u,u,u	u,u,u,u,u

Vuit instàncies enviades a classificar:

The set sent to classify:	The value predicted:	The precision computed for this unitary value:
high,low,5more,more,small,high,acc	**unacc**	**44,24**
vhigh,high,2,4,big,low,unacc	**unacc**	**22,67**
med,low,4,2,big,low,unacc	**acc**	**36,36**
med,high,2,more,small,low,unacc	**unacc**	**44,24**
low,low,2,more,big,med,good	**unacc**	**44,24**
vhigh,low,4,2,med,low,unacc	**unacc**	**35,95**
vhigh,vhigh,2,2,med,med,unacc	**unacc**	**28,76**
high,low,5more,4,small,high,acc	**unacc**	**44,24**
The final precision for the predictions set is: 50,00 %		

		D. Stump	J48	KNN	SMO
Agent 1 196 inst.	**Precision**	25%	25%	25%	37.5%
	Predicted Values	u,u,u,u,u,u,u,u	u,u,u,u,u,u,u,u	u,u,u,u,u,u,u,u	u,u,u,u,u,u,g,u
Agent 2 1036 inst.	**Precision**	25%	50%	50%	50%
	Predicted Values	u,u,u,u,u,u,u,u	u,u,a,u,u,u,a,u	u,u,a,u,u,u,a,u	u,u,a,u,u,u,a,u
Agent 3 1395 inst.	**Precision**	25%	50%	50%	62.5%
	Predicted Values	u,u,u,u,u,u,u,u	u,a,a,u,u,u,u,u	u,a,a,u,u,u,u,u	u,a,a,u,u,u,u,u

4.4. Output per 4 Agents Classificadors

Per tot l'apartat 4.4 següent, el sistema conté quatre agents classificadors. El nombre d'instàncies enviades a classificar en test varia tal i com es mostra a continuació:

Una instància enviada a classificar:

The set sent to classify	The value predicted	The precision computed for this unitary value
high,high,3,4,small,low,unacc	**unacc**	**100,00**
The final precision for the predictions set is: 100,00 %		

		D. Stump	J48	KNN	SMO
Agent 1 621 inst.	**Precision**	100%	100%	100%	100%
	Predicted Values	u	u	u	u
Agent 2 633 inst.	**Precision**	100%	100%	100%	100%
	Predicted Values	u	u	u	u
Agent 3 370 inst.	**Precision**	100%	100%	100%	100%
	Predicted Values	u	u	u	u
Agent 4 1628 inst.	**Precision**	100%	100%	100%	100%
	Predicted Values	u	u	u	u

Tres instàncies enviades a classificar:

The set sent to classify:	The value predicted:	The precision computed for this unitary value:
low,med,4,4,med,low,unacc	**unacc**	**96,82**
vhigh,high,4,more,small,low,unacc	**unacc**	**90,45**
vhigh,med,5more,2,med,med,unacc	**unacc**	**96,82**
The final precision for the predictions set is: 100,00 %		

		D. Stump	J48	KNN	SMO
Agent 1 621 inst.	**Precision**	100%	100%	100%	100%
	Predicted Values	u,u,u	u,u,u	u,u,u	u,u,u
Agent 2 633 inst.	**Precision**	100%	100%	66%	66%
	Predicted Values	u,u,u	u,u,u	u,a,u	u,a,u
Agent 3 370 inst.	**Precision**	100%	100%	100%	100%
	Predicted Values	u,u,u	u,u,u	u,u,u	u,u,u
Agent 4 1628 inst.	**Precision**	100%	100%	100%	100%
	Predicted Values	u,u,u	u,u,u	u,u,u	u,u,u

Cinc instàncies enviades a classificar:

The set sent to classify:	The value predicted:	The precision computed for this unitary value:
med,low,5more,2,big,med,unacc	**unacc**	**49,00**
med,low,2,4,small,med,acc	**unacc**	**49,00**
med,med,4,4,small,low,unacc	**unacc**	**49,00**
vhigh,med,2,more,small,low,unacc	**unacc**	**27,01**
low,low,2,4,big,high,vgood	**unacc**	**33,98**
The final precision for the predictions set is: 60,00 %		

		D. Stump	J48	KNN	SMO
Agent 1 621 inst.	**Precision**	60%	60%	60%	40%
	Predicted Values	u,u,u,u,u	u,u,u,u,u	u,u,u,u,u	u,u,u,v,u
Agent 2 633 inst.	**Precision**	60%	60%	60%	60%
	Predicted Values	u,u,u,u,u	u,u,u,u,u	u,u,u,u,u	u,u,u,u,u
Agent 3 370 inst.	**Precision**	60%	40%	40%	40%
	Predicted Values	u,u,u,u,u	u,u,u,a,u	u,u,u,a,u	u,u,u,a,u
Agent 4 1628 inst.	**Precision**	60%	40%	40%	40%
	Predicted Values	u,u,u,u,u	u,u,u,a,a	u,u,u,a,a	u,u,u,a,a

Vuit instàncies enviades a classificar:

The set sent to classify for this unitary value	The value predicted	The precision computed
low,vhigh,4,4,med,high,acc	**unacc**	**45,04**
high,high,4,more,small,low,unacc	**unacc**	**26,81**
vhigh,low,2,2,big,med,unacc	**acc**	**22,00**
low,vhigh,5more,4,big,med,acc	**unacc**	**45,04**
low,high,5more,2,big,high,unacc	**unacc**	**41,62**
vhigh,vhigh,3,4,small,low,unacc	**unacc**	**45,04**
high,low,2,2,small,med,unacc	**acc**	**29,77**
vhigh,high,3,2,big,high,unacc	**unacc**	**26,95**
The final precision for the predictions set is: 50,00 %		

		D. Stump	J48	KNN	SMO
Agent 1 621 inst.	**Precision**	75%	62.5%	62.5%	62.5%
	Predicted Values	u,u,u,u,u,u,u,u	u,u,u,u,a,u,u,u	u,u,u,u,a,u,u,u	u,u,u,u,a,u,u,u
Agent 2 633 inst.	**Precision**	75%	62.5%	62.5%	37.5%
	Predicted Values	u,u,u,u,u,u,u,u	u,u,u,u,a,u,u,u	u,u,u,u,a,u,u,u	u,u,a,u,a,u,a,u
Agent 3 370 inst.	**Precision**	75%	75%	75%	62.5%
	Predicted Values	u,u,u,u,u,u,u,u	u,u,u,u,u,u,u,u	u,u,u,u,u,u,u,u	u,u,u,u,a,u,u,u
Agent 4 1628 inst.	**Precision**	75%	75%	75%	62.5%
	Predicted Values	u,u,u,u,u,u,u,u	u,u,u,a,u,u,u,u	u,u,u,a,g,u,u,u	u,u,u,a,g,u,u,u

4.5. Output per 5 Agents Classificadors

Per tot l'apartat 4.5 següent, el sistema conté cinc agents classificadors. El nombre d'instàncies enviades a classificar en test varia tal i com es mostra a continuació:

Una instància enviada a classificar:

The set sent to classify	The value predicted	The precision computed for this unitary value
low,low,3,2,med,high,unacc	**unacc**	**100,00**
The final precision for the predictions set is: 100,00 %		

		D. Stump	J48	KNN	SMO
Agent 1 1474 inst.	**Precision**	100%	100%	100%	100%
	Predicted Values	u	u	u	u
Agent 2 1718 inst.	**Precision**	100%	100%	100%	100%
	Predicted Values	u	u	u	u
Agent 3 1431 inst.	**Precision**	100%	100%	100%	100%
	Predicted Values	u	u	u	u
Agent 4 1519 inst.	**Precision**	100%	100%	100%	100%
	Predicted Values	u	u	u	u
Agent 5	**Precision**	100%	100%	100%	100%

1652 inst.	**Predicted Values**	u	u	u	u

Tres instàncies enviades a classificar:

The set sent to classify for this unitary value	The value predicted	The precision computed
med,med,2,4,big,low,unacc	**unacc**	**98,20**
high,vhigh,2,more,big,low,unacc	**unacc**	**94,70**
high,med,5more,2,small,low,unacc	**unacc**	**98,20**
The final precision for the predictions set is: 100,00 %		

		D. Stump	J48	KNN	SMO
Agent 1 1474 inst.	**Precision**	100%	100%	100%	100%
	Predicted Values	u,u,u	u,u,u	u,u,u	u,u,u
Agent 2 1718 inst.	**Precision**	100%	100%	100%	100%
	Predicted Values	u,u,u	u,u,u	u,u,u	u,u,u
Agent 3 1431 inst.	**Precision**	100%	100%	100%	100%
	Predicted Values	u,u,u	u,u,u	u,u,u	u,u,u
Agent 4 1519 inst.	**Precision**	100%	100%	66.6%	100%
	Predicted Values	u,u,u	u,u,u	u,a,u	u,u,u
Agent 5 1652 inst.	**Precision**	100%	100%	100%	100%
	Predicted Values	u,u,u	u,u,u	u,u,u	u,u,u

Cinc instàncies enviades a classificar:

The set sent to classify:	The value predicted:	The precision computed for this unitary value
low,high,3,more,med,med,acc	**acc**	**56,68**
vhigh,low,5more,more,med,high,acc	**acc**	**56,68**
low,high,4,2,med,high,unacc	**acc**	**50,37**
vhigh,low,4,more,small,med,unacc	**acc**	**56,68**
low,vhigh,5more,4,big,med,acc	**acc**	**46,88**
The final precision for the predictions set is: 60,00 %		

		D. Stump	J48	KNN	SMO
Agent 1 406 inst.	**Precision**	60%	40%	40%	40%
	Predicted Values	a,a,a,a,a,	a,a,a,a,u	a,a,a,a,u	a,a,a,a,u
Agent 2 207 inst.	**Precision**	60%	60%	60%	40%
	Predicted Values	a,a,a,a,a,	a,a,a,a,a,	a,a,a,a,a,	a,a,a,a,u
Agent 3 1097 inst.	**Precision**	60%	40%	40%	40%
	Predicted Values	a,a,a,a,a,	a,a,a,a,u	a,a,a,a,u	a,a,a,a,u
Agent 4 1519 inst.	**Precision**	60%	80%	60%	60%
	Predicted Values	a,a,a,a,a,	a,a,u,a,a,	a,a,a,a,a,	a,a,a,a,a,
Agent 5 1587 inst.	**Precision**	60%	60%	60%	60%
	Predicted Values	a,a,a,a,a,	a,a,a,a,a,	a,a,a,a,a,	a,a,a,a,a,

Vuit instàncies enviades a classificar:

The set sent to classify:	The value predicted:	The precision computed for this unitary value:
vhigh,high,5more,2,big,high,unacc	**unacc**	**76,31**
low,low,2,2,small,high,unacc	**unacc**	**57,14**
med,high,2,4,med,med,unacc	**unacc**	**66,89**
med,high,3,2,big,low,unacc	**unacc**	**71,04**
low,med,2,2,small,low,unacc	**unacc**	**60,04**
low,vhigh,2,4,med,high,acc	**unacc**	**64,39**
vhigh,low,2,2,small,low,unacc	**unacc**	**71,63**
med,high,2,2,big,low,unacc	**unacc**	**71,04**
The final precision for the predictions set is: 87,50 %		

		D. Stump	J48	KNN	SMO
Agent 1 1007 inst.	**Precision**	87.5%	87.5%	50%	37.5%
	Predicted Values	u,u,u,u,u,u,u,u	u,u,u,u,u,u,u,u	u,u,u,a,a,u,u,a	u,u,u,a,a,u,a,a
Agent 2 1369 inst.	**Precision**	87.5%	87.5%	75%	75%
	Predicted Values	u,u,u,u,u,u,u,u	u,u,u,u,u,u,u,u	u,a,u,u,a,a,u,u	u,a,u,u,a,a,u,u
Agent 3 1484 inst.	**Precision**	87.5%	87.5%	75%	87.5%
	Predicted Values	u,u,u,u,u,u,u,u	u,u,u,u,u,u,u,u	u,a,a,u,u,a,u,u	u,u,u,u,u,u,u,u
Agent 4 1393 inst.	**Precision**	87.5%	87.5%	62.5%	75%
	Predicted Values	u,u,u,u,u,u,u,u	u,u,u,u,u,u,u,u	u,a,a,u,u,u,u,u	u,a,u,u,u,u,u,u

Agent 5 1386 inst.	**Precision**	87.5%	87.5%	62.5%	62.5%
	Predicted Values	u,u,u,u,u,u,u,u	u,u,u,u,u,u,u,u	u,u,a,a,a,u,u,a	u,u,u,u,a,u,a,u

4.6. Comprovació dels resultats obtinguts

A continuació, es mostra com s'han obtingut el valors finals de predicció i la precisió unitària associada.

El valor predit final es determina com:

$$\boldsymbol{MAX}\left(\sum_{f(i),g(j)}\{\boldsymbol{v_m}:\ \boldsymbol{p_{ij}w_i}\}\right)\ \textit{for each different } v_m$$

On p_{ij} representa la precisió de cada algorisme de cada agent i v_m representa cada possible valor predit que podem obtenir.
La funció f(i,j) és una funció que retorna els 'i' (agents) i els 'j' (algorismes) tals que el valor predit sigui el mateix.

Si agafem una execució a l'atzar (i marcada en roig l'instància que volem estudiar):

The set sent to classify:	The value predicted:	The precision computed for this unitary value:
high,low,5more,more,small,high,acc	**unacc**	**44,24**
vhigh,high,2,4,big,low,unacc	**unacc**	**22,67**
med,low,4,2,big,low,unacc	**acc**	**36,36**
med,high,2,more,small,low,unacc	**unacc**	**44,24**
low,low,2,more,big,med,good	**unacc**	**44,24**
vhigh,low,4,2,med,low,unacc	**unacc**	**35,95**
vhigh,vhigh,2,2,med,med,unacc	**unacc**	**28,76**
high,low,5more,4,small,high,acc	**unacc**	**44,24**
The final precision for the predictions set is: 50,00 %		

		D. Stump	**J48**	**KNN**	**SMO**
Agent 1 196 inst.	**Precision**	25%	25%	25%	37.5%
	Predicted Values	u,u,**u**,u,u,u,u,u	u,u,**u**,u,u,u,u,u	u,u,**u**,u,u,u,u,u	u,u,**u**,u,u,u,g,u
Agent 2 1036 inst.	**Precision**	25%	50%	50%	50%
	Predicted Values	u,u,**u**,u,u,u,u,u	u,u,**a**,u,u,u,a,u	u,u,**a**,u,u,u,a,u	u,u,**a**,u,u,u,a,u
Agent 3 1395 inst.	**Precision**	25%	50%	50%	62.5%
	Predicted Values	u,u,**u**,u,u,u,u,u	u,a,**a**,u,u,u,u,u	u,a,**a**,u,u,u,u,u	u,a,**a**,u,u,u,u,u

Els possibles valors finals són: {u,a}

V_1='u'=$\frac{(25*196+25*196+25*196+37,5*196)+(25*1036)+(25*1395)}{4*(196+1036+1395)} = \frac{22050+25900+34875}{10508}$=7.88

V_2= 'a'= $\frac{(50*1036+50*1036+50*1036)+(50*1395+50*1395*62,5*1395)}{4*(196+1036+1395)} = \frac{155400+226687,5}{10508}$ = 36,36

Per tant, el valor predit és:

MAX ({'u' : 7.88} , {'a' : 36,36}) = {'a' : 36,36} → 'a' amb precisió unitària 36,36

La precisió global del càlcul ve donada com:

P = $\frac{instàncies\ correctes}{Total\ Instàncies}$ = $\frac{4}{8}$ = 0.50 = 50.0%

4.7. Anàlisi de les proves

En cada prova, independentment del nombre d'agents del sistema, cada agent s'entrena amb el seu conjunt pertinent de dades *trainN.csv* (que podríem anomenar, també, servidor privat). A més cada agent, executa les prediccions per el conjunt test (enviat pel Manager) amb els quatre algorismes supervisats que té allotjats.

Així doncs, com que cada agent s'entrena amb un conjunt independent diferent de dades (*trainN.csv*), implica que les prediccions amb cada un dels quatre algorismes supervisats que té allotjats (i atès que cada algorisme té diferents tècniques de predicció), puguin ser diferents.

Per tant, si tenim N agents en el sistema (entrenats cada un amb els seu *trainN* pertinent) i cada agent té allotjats quatre algorismes supervisats de predicció (*SMO, J48, KNN i Decision Stump*) podem obtenir, en total, 4*N classificacions diferents.

Atès que el Manager rep 4*N classificacions, ha de computar d'entre totes les prediccions rebudes, el millor valor per cada instància del test enviat a classificar. Per realitzar aquest còmput, el Manager, escull aquell valor, d'entre el 4*N rebuts, tal que, les sumes de raons entre precisions i nombre d'instàncies sigui més gran.

Si analitzem els resultats obtinguts s'observa que el comportament, en general, és més precís per a la classificació de conjunts test quan aquest té un nombre baix d'instàncies. La explicació trobada és la següent:

Si analitzem les dades de *car.csv*, trobem que hi ha 1732 instàncies de les quals, les classes finals de les instàncies poden variar en quatre valors *(unacc, acc, vgood, good*). A més s'observa un clar predomini en instàncies de valor final *unacc* (per tant, més probables de ser escollides en la funció aleatòria que genera les instàncies de *test.csv*).

A més, s'observa que hi ha combinacions d'atributs, en les què principalment, predomina una classe final, i hi ha d'altres combinacions d'atributs molt semblants en les que hi ha dos o més classes finals diferents.

Per tant, per aquelles combinacions d'atributs en què predomina una classe final l'algorisme supervisat predirà aquella classe final i, per aquelles combinacions en què poden haver-hi dos o més classes finals (intersecció de combinacions d'atributs amb diferents classes finals), predirà aquella classe que tingui major nombre d'instàncies en el conjunt *train*.

Si ho traslladem a les sortides de programari, es veu que per baix nombre d'instàncies (1,3), on la gran majoria de les combinacions d'atributs generades per classificar són de classe final *unacc* (que són les que tenen més probabilitat de ser escollides en la funció *random* que genera *test.csv*), sempre classifica bé amb tendència al 100% de precisió.

En canvi, per conjunts amb més instàncies (5,8), on la funció *random*, escull combinacions d'atributs que tenen diferent classe final, l'algorisme supervisat, per aquelles combinacions d'atributs que poden ser la intersecció de dues classes finals diferents, escull aquella classe de la que hi ha més nombre d'instàncies.

Si portem a la pràctica aquest últim fet, podem dir que algunes combinacions d'atributs en test que no tenen classe final la més predominant (classes com: *acc, good, vgood*) però que pertanyen a combinacions d'atributs que *intersecten* amb les combinacions d'atributs de classe final unacc (sempre major en nombre), classifiquin com *unacc* aquella combinació d'atributs i que, per tant, disminueixi la precisió al generar error.

Així doncs, es pot atribuir a aquest procediment, l'error en la predicció final en sortides de moltes instàncies, en les què principalment es classifiquen instàncies com *unacc* quan pertanyen a d'altres classes. Tot i així, també hi poden haver interseccions d'atributs en què la major afluència sigui la classe *acc* i predigui una classe *unacc* com *acc*.

Finalment si analitzem els classificadors, en tant que nombre d'instàncies, sabem que, en general, per conjunts d'entrenament amb elevat nombre d'instàncies, la precisió de les prediccions sempre és major. En aquest cas, en les sortides del programari i, per classificadors amb poques instàncies, no es nota diferència en la precisió.

Es pot atribuir aquest fet a que, tot i que, l'agent tingui poques combinacions d'atributs en el seu entrenament, tingui suficient exemples de cada combinació per predir el mateix que els classificadors amb alts volums d'entrenament.

4.8. Intercanvi de missatges. *Sniffer*

En la següent imatge es pot veure el funcionament del programari mitjançant l'ús *d'sniffers* (integrats en el entorn JADE). Obeeix la cronologia de les següents comandes introduïdes per l'usuari:

- creació de quatre agents amb el pertinent càlcul de prediccions.
- dos ordres per llistar els classificadors *online*.
- creació d'un sol agent.
- l'enviament de noves dades a classificar.

Es pot apreciar també, els tipus d' *ACLMessage* utilitzats en el programari. Quan els agents classificadors estan a punt per classificar, envien un *INFORM* al *Manager*. Quan el Manager envia dades per classificar als classificadors envia missatges de tipus *PROPOSE* i, finalment, quan els classificadors envien resultats utilitzen un missatge del tipus *REQUEST*.

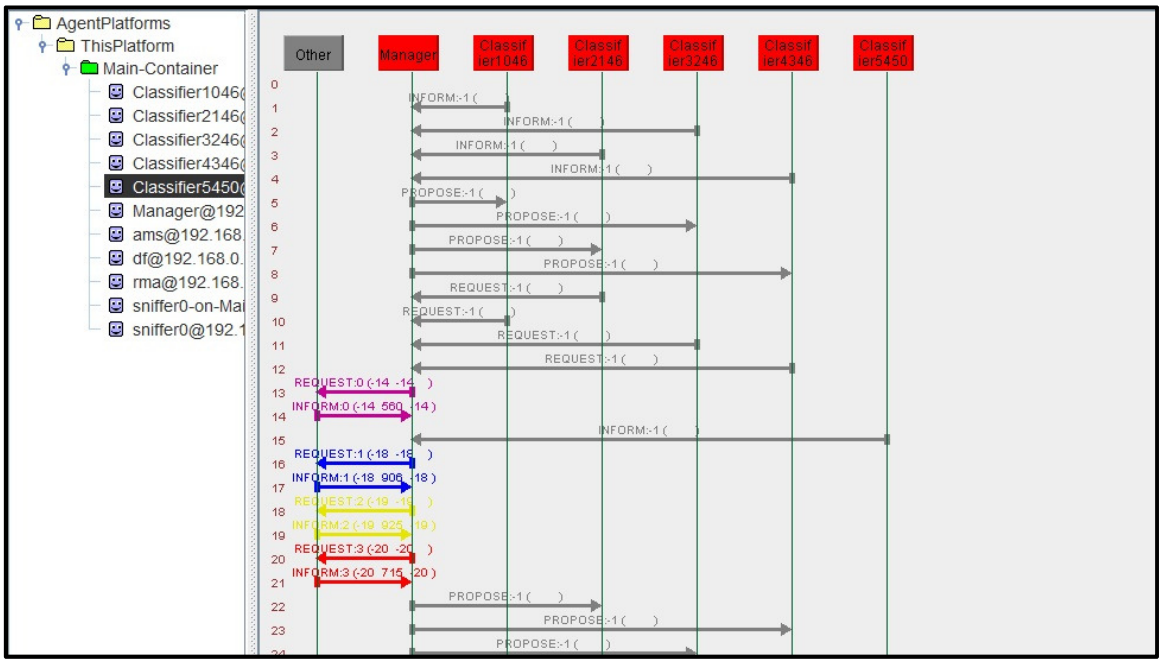

Figura 16: Sniffer

5. Conclusions

La primera conclusió que es pot treure, és que amb la planificació calculada en el pla de treball s'ha aconseguit acabar el projecte a temps. Hi ha alguna prova, com és la comprovació del TFG amb el programari Weka que es deixa per a possibles modificacions del programari.

S'ha implementat un programari en què s'ha creat diferents agents amb diferents conjunt de dades per l'entrenament que emulen perfectament el que podria ser un cas real en quant a servidors distribuïts i privats.

Algunes proves sobre el programari mostren que, atès que el *Manager* està preparat per rebre tantes conclusions com agents Classificadors creats, si un agent Classificador falla, el sistema es queda penjat a l'espera dels resultats de l'agent Classificador que ha fallat. Podria ser un tema a millorar en futures modificacions.

S'ha decidit també, limitar el nombre d'agents a cinc i el nombre d'instàncies a nou (que encara es considera elevat), ja que, s'ha comprovat que per nombres alts d'agents i d'instàncies, la probabilitat d'errors en execució augmenta. Per tant, tot i que augmentar el límit d'agents i casos a predir seria una variació d'implementació molt senzilla, seria recomanable reduir-los per tal de fer el programari més robust.

Cada agent té allotjat una granja d'algorismes supervisats, per tant, cada agent té molt força en la predicció de dades. A més, després de valorar les diferents classificacions rebudes, s'ha decidit generar com resultat final el valor que té més força, vers el nombre d'instàncies i precisió de cada algorisme de cada agent classificador.

En qüestions d'implementació, per trobar errades, s'ha decidit prescindir del *debugger* de l'Eclipse atès, el gran nombre de classes que té el programari. Per buscar errades s'ha decidit escriure per pantalla.

També, per qüestions de disseny s'han creat tots els agents en el contenidor principal, encara que, es podrien haver creat en contenidors diferents. El programari estaria preparat, amb petites modificacions, per treballar en entorns distribuïts.

Finalment i, de caire personal, s'han complert les expectatives de la primera PAC on es definien tots els objectius d'aprenentatge. S'han utilitzat coneixements de disseny UML, com diagrames de seqüència, programació Java, entorn Jade, etc.

6. BIBLIOGRAFIA

1. Bellifemine Fabio, Caire, Giovani, Greenwood, Dominic. *Developing multi-agent system. With Jade.* .Chichester: Wiley, 2007. 286p. ISBN: 978-0-470-05747-6 (HB).

2. Coll Corbilla, Jordi. *Sistema de presa de decisions distribuït*. [consulta en línia] Barcelona: UOC 2014. 93 p. http://openaccess.uoc.edu/webapps/o2/handle/10609/32761 [data consulta 18/10/14]

3. Isern Alarcón, David, Sánchez Ruenes, David [consulta en línia]. *Guia de programació de Sistemes Multiagent en JADE 3.3*. Universitat Rovira i Virgili. 8 de Juny de 2005. 56 p. http://deim.urv.cat/recerca/reports/DEIM-RT-05-001.html [data consulta 12/10/14]

4. *Tutorial2: Starting Jade*. [consulta en línia]. http://jade.tilab.com/doc/tutorials/JADEAdmin/startJade.html [data consulta: 15/11/14]

5. *Use WEKA in your Java code.* [consulta en lína]. http://weka.wikispaces.com/Use+WEKA+in+your+Java+code [data consulta: 3/11/14]

7. ANNEX

7.1. Instruccions d'ús del programari

Les instruccions estan explicades per executar en un entorn Windows.
El primer pas és desar la carpeta TFG en l'arrel del disc: "C:\TFG"

7.1.1. Execució en l'entorn Eclipse

1. Desar la carpeta TFG en la carpeta *workspace*.
2. Crear un nou projecte en la consola del *Eclipse* anomenada TFG.
3. Escriure les següents comandes marcades en roig en la pestanya *Run/Run Configurations* de *l'Eclipse*.

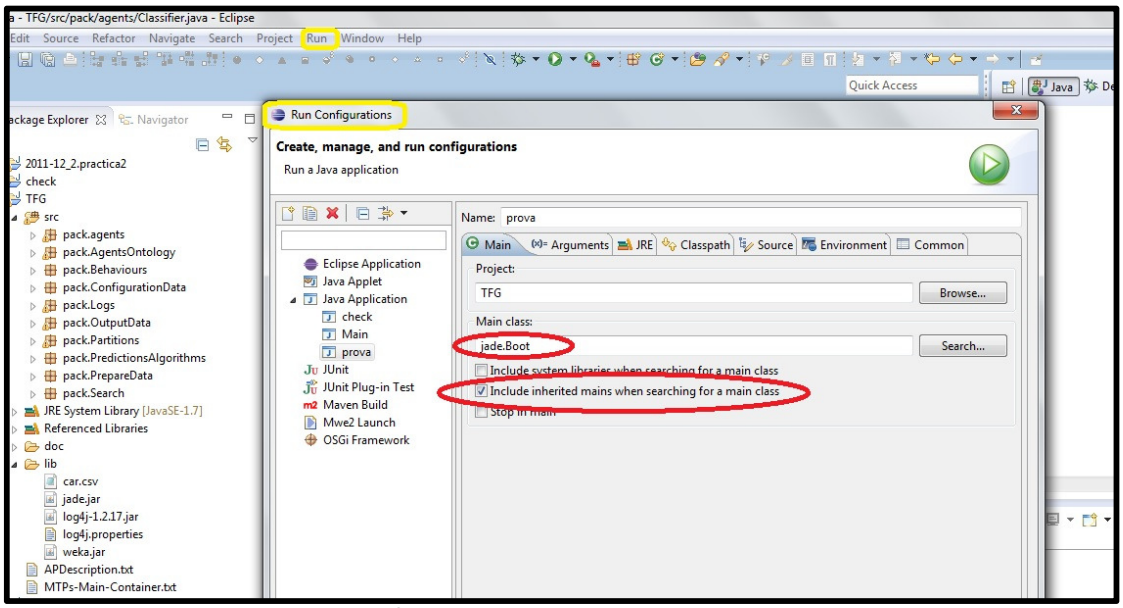

Figura 17: Entorn Eclipse: Run Configurations

4. Escriure els següents arguments (ressaltats en roig en l'imatge):
   -gui –agents Manager:pack.agents.Manager –host localhost –local-port 1099

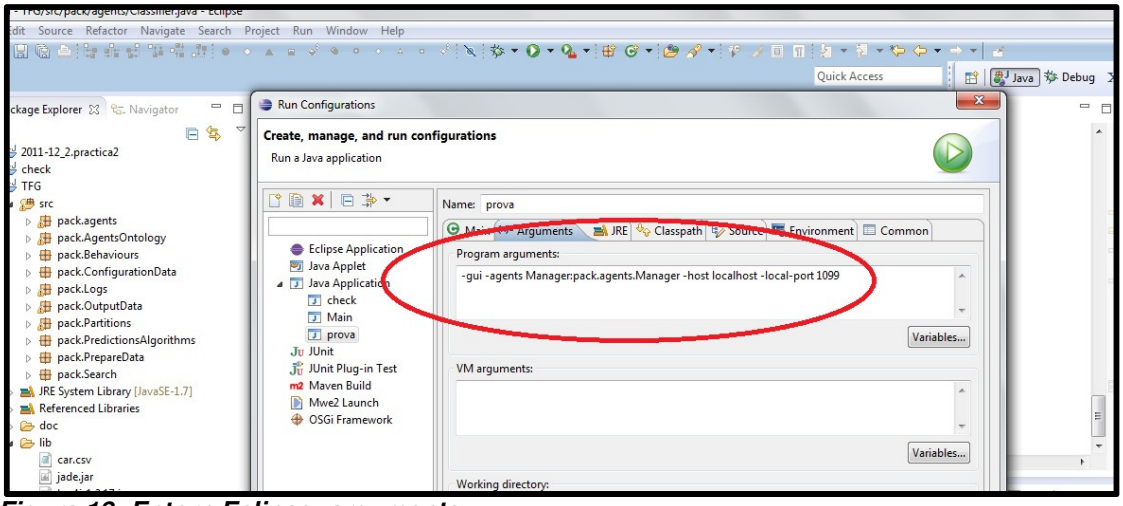

Figura 18: Entorn Eclipse: arguments

5. Prémer el botó dret del ratolí sobre el projecte TFG i seguir el ressaltat de la següent imatge:

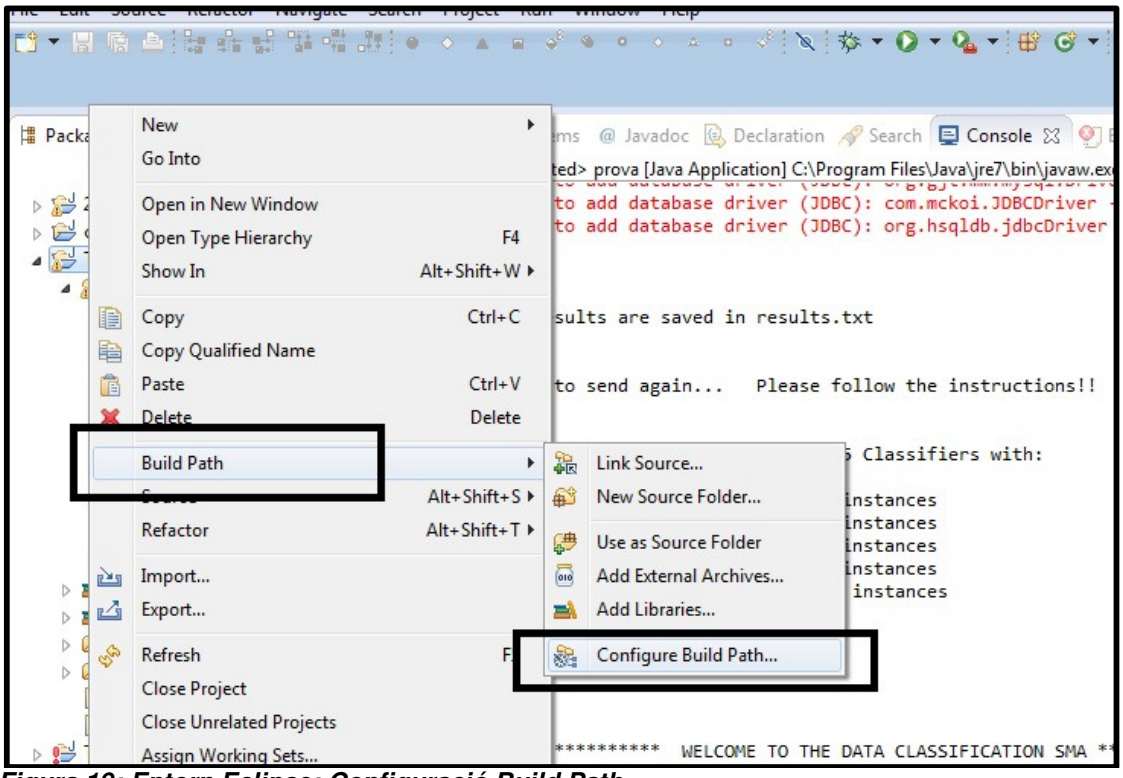

Figura 19: Entorn Eclipse: Configuració Build Path

6. Afegir les classes *weka.jar, jade.jar i log4j-1.2.17.jar* com a llibreries externes.
   Es troben en el *path*: C:\TFG\lib\

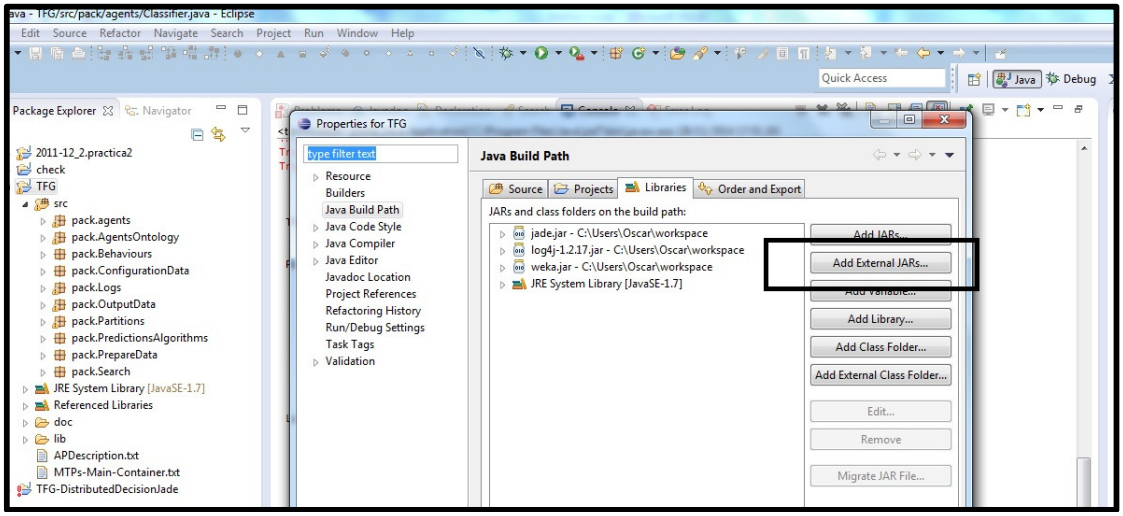

Figura 20: Entorn Eclipse: Afegir .jar externs

7. Executar.
   Les sortides generades pel programa (trainN.*csv, test.csv results.txt* i *log.txt*) es crearan en la carpeta lib del projecte TFG. La ruta per trobar-los és: "C:\TFG\lib"

7.1.2. Execució en la consola

Per executar el programari en la consola de Windows, executar els següent passos:

1. Anar a la ruta → C:\TFG\bin

2. Modificar la variable CLASSPATH amb la següent comanda:
   C:\TFG\bin> **set CLASSPATH=%CLASSPATH%;.;C:\TFG\lib\jade.jar;C:\TFG\lib\weka.jar; C:\TFG\lib\log4j-1.2.17.jar;C:\TFG\lib**

3. Executar la comanda:

C:\TFG\bin> **java jade.Boot –gui –agents Manager:pack.agents.Manager** *–host localhost –port-local 1099 (opcional pel port i la màquina)*

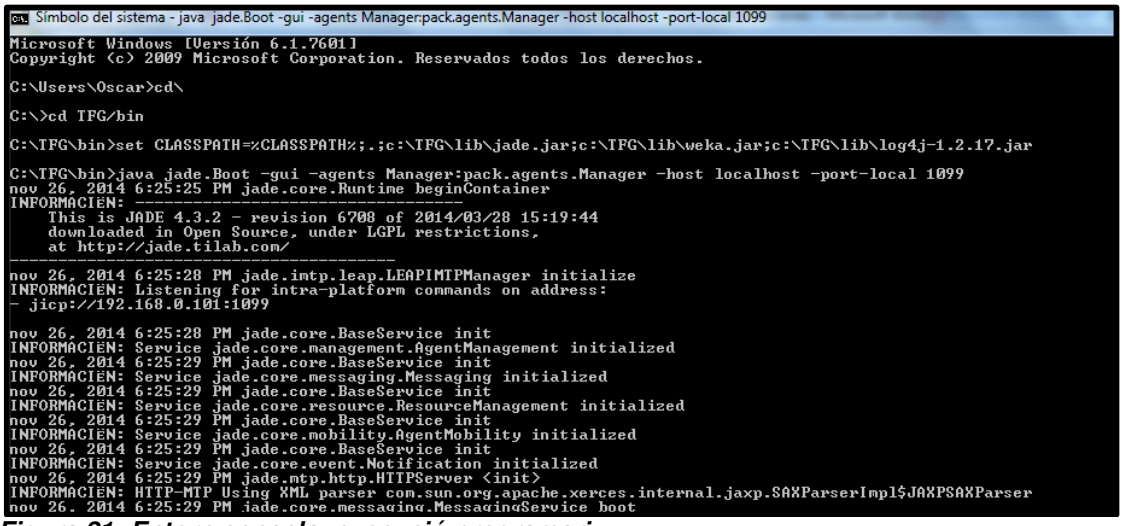

Figura 21: Entorn consola: execució programari

7.1.3. Execució del fitxer .bat

Executar l'arxiu *runSoftware.bat* (annex al programari en la carpeta TFG/Execute/). Prèviament, la carpeta TFG ha d'estar desada en l'arrel del disc (C:\) en entorns Windows.

7.2. Enllaços al programari utilitzat

En els següents enllaços es poden trobar els webs on descarregar el programari utilitzat en el desenvolupament d'aquest treball.

- Eclipse Juno:
  https://eclipse.org/downloads/packages/release/Juno/SR2

- Java JDK7:
  http://www.oracle.com/technetwork/es/java/javase/downloads/index.html

- Jade (v:4.3.2) (es pot trobar el jade.jar en la carpeta /bin/)
  http://jade.tilab.com/download/jade/license/jade-download/?x=31&y=17

- Weka (v:weka-stable-3.6.6.jar): http://mvnrepository.com/artifact/nz.ac.waikato.cms.weka/weka-stable/3.6.6

- Log4j-1.2.17.jar: https://archive.apache.org/dist/logging/log4j/1.2.17/

7.3. Diagrama UML complet

A continuació es pot veure el diagrama UML de totes les classes del programari i, amb els seus mètodes principals:

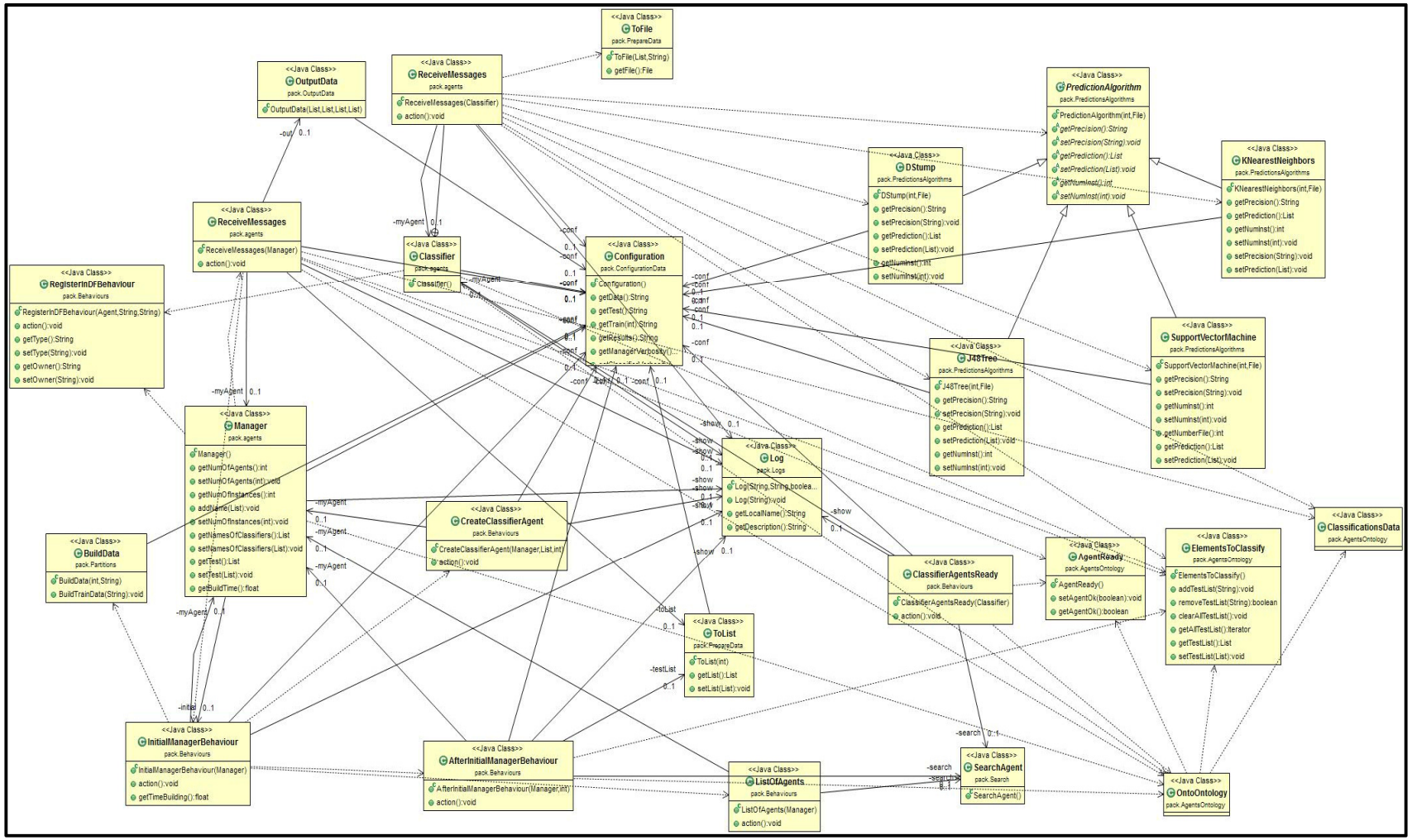

Figura 22: Diagrama UML complet