"Sistema de presa de decisions distribuït".

Alumne: Jordi Coll i Corbilla
Grau d'Enginyeria en Informàtica

Consultor: David Isern Alarcón

Data Lliurament: 11 de juny del 2014

Índex

Índex de figures

1. Resum

En aquest treball es pretén realitzar una aplicació que combina **agents intel·ligents** i **algorismes tradicionals** d'Intel·ligència artificial tals com algorismes classificadors. La idea central d'aquest projecte és la de crear un **sistema multi-agent que permeti confrontar diverses opinions en una única**. Una idea similar és la que es va utilitzar en el projecte *HealthAgents*[1], un multi agent distribuït composat d'agents intel·ligents amb un sistema de decisió per la correcta classificació de tumors cerebrals implementant patrons de reconeixent i mètodes de discriminació sobre espectroscòpies de ressonància magnètica.

Una estructura genèrica d'aquesta arquitectura es pot veure en la següent figura:

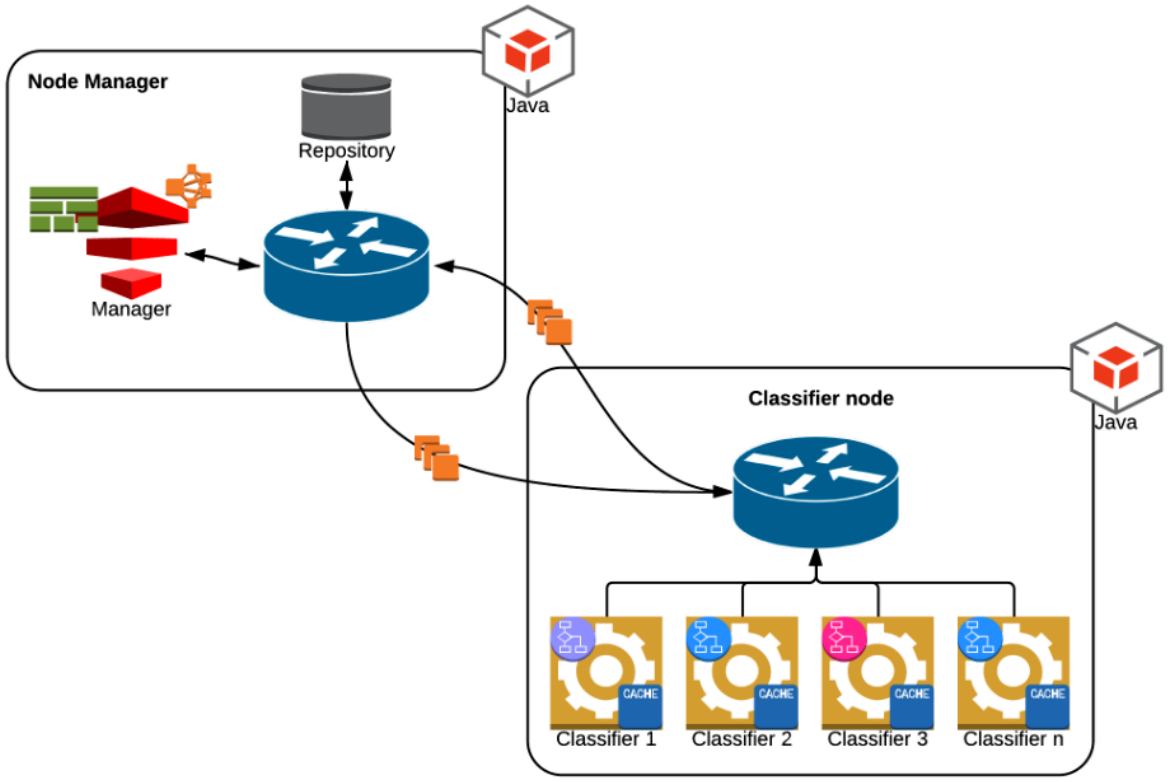


Figura 1 Arquitectura multi-node del sistema de presa de decisions distribuït

En general, aquest projecte vol combinar diferents **agents** que utilitzin algorismes **classificadors** localment i que posteriorment enviaran les seves respostes a un **agent central** que serà capaç de **composar** la idea general que sense l'ajuda dels agents distribuïts no hagués pogut ser possible. La idea central d'aquest projecte és la creació d'aquesta arquitectura mitjançant **JADE** i **Weka** i aplicar-la a un problema real.

2 Introducció

L'objectiu d'aquest apartat és iniciar l'exposició del projecte indicant els motius pel que s'ha triat la matèria dels sistemes multi agent com a base del treball i quins són a grans trets els objectius que es volen assolir. També s'indiquen les fases del treball i planificació que es seguiran durant tot el projecte.

2.1 Justificació i motivació del TFG

Des de que em vaig iniciar en el món de la Intel·ligència artificial, vaig quedar fascinat amb el paradigma dels agents intel·ligents. Els SMA (Sistemes MultiAgent) son conjunts d'entitats capaces de treballar d'una manera coordinada per a la resolució d'un problema concret i el fet de poder posar en pràctica tots els coneixements apresos durant els estudis del Grau en un projecte com aquest em motiven moltíssim.

Els Sistemes multiagent ens ofereixen la possibilitat de poder crear *universos artificials*[2] que realment funcionen com a petits laboratoris per comprovar teories de comportaments locals i ens ofereixen una alternativa a la solució de problemes d'una manera distribuïda ja que aquests es poden distribuir entre els agents i tornar-se molt més eficients i experts.

En l'assignatura d'aprenentatge computacional vaig poder aprofundir amb diferents algorismes classificadors i la idea de poder realitzar un SMA intel·ligent amb nodes capaços de classificar informació i aplicar aquesta arquitectura a un problema real és prou interessant com per a poder dur-la a terme com a TFG.

2.2 Objectius del TFG

L'objectiu general d'aquest projecte preveu l'estudi, disseny i implementació d'un sistema multiagent intel·ligent amb classificadors genèrics que proveeixen informació crucial per a la decisió final sobre problemes reals. En concret es simularà un sistema que permetrà classificar informació de qualsevol àmbit. Aquestes dades s'avaluaran a través dels classificadors que utilitzaran informació enviada per el agent central (manager) i que desaran en uns repositoris locals per la tasca de classificació.

Un cop les dades han sigut processades, aquestes s'enviaran a l'agent central (manager) que acabarà definint les dades de cada classificador. El manager que rep les dades genèriques, demana les dades al repositori i les agrega fent així un excel·lent exercici d'IA distribuïda.

Caldrà doncs, realitzar un estudi detallat sobre les tecnologies a utilitzar i sobretot la manera d'implementar els magatzems i repositoris. Aprofundir en les funcionalitats que ofereix la llibreria JADE per a la creació dels agents i la llibreria Weka per a la part dels classificadors i altres per la definició de l'ontologia.

2.2.1 Objectiu General

El següent projecte es centrarà en els següents objectius generals:

- Revisar materials dels SMA i teoria necessària.
- Estudiar la llibreria JADE per al desenvolupament dels agents. Familiaritzar-se amb aquests i desenvolupar un POC (Proof of concept).
- Disseny de l'arquitectura dels agents base: obtenció de les dades dels repositoris (https://archive.ics.uci.edu/ml/datasets.html), configuració d'aquests, algorisme classificador, dades d'entrenament.
- Disseny de l'arquitectura de l'agent manager: recopilació de la informació i presentació d'aquesta.
- Implementació del sistema utilitzant llibreries JADE.
- Implementació dels algorismes classificadors mitjançant llibreries Weka.
- Aplicació i resolució d'un problema concret, en aquest cas es prendrà un exemple del repositori UCI i s'executarà dins el sistema. Es definiran les dades a analitzar i mitjançant els agents classificadors es retornarà una resposta final a l'usuari.
- Jocs de proves i estudi dels resultats: cal verificar que les dades han sigut correctament analitzades i classificades.

2.2.2 Objectiu Específic

- Es disposarà d'un exemple del qual es vol obtenir informació. Des de el repositori UCI (https://archive.ics.uci.edu/ml/datasets.html) es seleccionarà un o varis exemples i s'analitzaran dins el sistema.
    - Les dades caldrà formatar-les per a que els atributs estiguin en la forma correcta i el que el Weka no tingui problemes per a fer la classificació.
- El manager, enviarà aquest elements als agents classificadors.
    - El manager obtindrà les dades del repositori. Si el repositori conté 1000 registres, aquest enviarà les dades a cada classificador utilitzant una certa distribució. Dins del paquet enviat, s'informarà al client sobre el registres que s'utilitzaran per a l'entrenament i els registres que s'utilitzaran per a la classificació. Per tant el manager pot decidir que 900 d'aquests registres seran per entrenar els classificadors i la resta seran els registres per fer la prova de classificació.
- Cada classificador confronta l'element rebut amb la seva base de dades d'entrenament i retorna la seva resposta.
    - La repartició de les dades pot ser disjunta, és a dir, cada classificador pot rebre un número diferent de registres per crear la seva base de dades d'entrenament. Això crearà classificadors que tindran més elements per entrenar versus altres que no i la seva resposta també serà més eficient que la resta d'agents.
    - Aquestes polítiques es definiran a nivell de classificador i s'analitzarà el rendiment en cada cas.
    - Es pretén crear un sistema dinàmic.
- El manager ha d'agregar tota la informació i composar una resposta final a l'usuari.
    - Els clients realment realitzen una funció lògica, reben uns paràmetres d'entrada i donen una sortida. El manager analitzarà la sortida de cada agent i donarà un resultat final sobre les dades.
- Cada Agent pot usar un mateix classificador o no per classificar.
    - Inicialment cada classificador utilitzarà el mateix classificador i s'anirà variant a mida que el projecte avanci.

- Cal definir un sistema per l'agregació de la informació tenint un compte uns pesos ja que cada agent pot tenir un criteri diferent que pot fer canviar la decisió final. Sistemes d'agregació podrien ser el OWA[3], WOWA[4], mitja ponderada, etc.
- El sistema multi-agent ha de crear una mena de vocabulari comú (ontologia) per tal que s'entenguin. Caldrà doncs, crear un petit vocabulari per tal de demanar una classificació i enviar uns resultats que continguin dades sense una mida fixa.

2.3 Estructuració del treball

El projecte estableix una planificació acurada d'una sèrie de tasques a assolir. Els següents apartats mostren la planificació i execució del treball definint una sèrie de tasques i objectius a aconseguir durant el període que dura el projecte així com la càrrega que m'aportarà. També es mostra el diagrama de Gantt amb les diferents tasques a través del temps.

2.4 Planificació i execució del treball

Es detalla a continuació la relació de les tasques i la seva planificació temporal.

2.4.1 Relació de tasques

La relació de les tasques del projecte és la següent i que es seguirà durant aquest:

	Task Mode	Task Name	Duration	Start	Finish	Predecessors
1		**Pla de treball**	**10 days**	**Thu 06/03/14**	**Fri 14/03/14**	
2		Pre-estudi tecnologies (SAM, JADE)	4 days	Thu 06/03/14	Sun 09/03/14	
3		Descripció funcionalitats	2 days	Sun 09/03/14	Mon 10/03/14	2
4		Objectius	2 days	Mon 10/03/14	Wed 12/03/14	3
5		Planificació	2 days	Wed 12/03/14	Fri 14/03/14	4
6		**Estudi I anàlisi del problema**	**31 days**	**Fri 14/03/14**	**Mon 07/04/14**	
7		Agent base genèric	6 days	Fri 14/03/14	Tue 18/03/14	5
8		Comportaments	5 days	Tue 18/03/14	Sat 22/03/14	7
9		Inferència	5 days	Sat 22/03/14	Wed 26/03/14	8
10		Algorisme classificador	5 days	Wed 26/03/14	Sun 30/03/14	9
11		Comunicació agents	5 days	Sun 30/03/14	Thu 03/04/14	10
12		Aplicació específica	5 days	Fri 04/04/14	Mon 07/04/14	11
13		**Requeriments I arquitectura**	**19 days**	**Mon 07/04/14**	**Tue 22/04/14**	
14		Disseny agent genèric	4 days	Mon 07/04/14	Fri 11/04/14	12
15		Disseny agent col·lector	3 days	Fri 11/04/14	Sun 13/04/14	14
16		Classes de configuració	3 days	Sun 13/04/14	Tue 15/04/14	15
17		Interfície I control de proves	3 days	Tue 15/04/14	Fri 18/04/14	16
18		Marc genèric d'aplicació	3 days	Fri 18/04/14	Sun 20/04/14	17
19		Marc específic d'aplicació	3 days	Sun 20/04/14	Tue 22/04/14	18
20		**Implementació**	**23 days**	**Tue 22/04/14**	**Sun 11/05/14**	
21		Agents (distribuïts I col·lector)	10 days	Tue 22/04/14	Wed 30/04/14	19
22		Sistema configurable	8 days	Thu 01/05/14	Wed 07/05/14	21
23		Logs I monitorització	5 days	Wed 07/05/14	Sun 11/05/14	22
24		**Joc de proves**	**12 days**	**Sun 11/05/14**	**Tue 20/05/14**	
25		Marc genèric	6 days	Sun 11/05/14	Fri 16/05/14	23
26		Marc Específic	6 days	Fri 16/05/14	Tue 20/05/14	25
27		**Confecció d'entregues**	**12 days**	**Tue 20/05/14**	**Fri 30/05/14**	
28		Documentació	2 days	Tue 20/05/14	Thu 22/05/14	26
29		Codi font + binaris	1 day	Thu 22/05/14	Fri 23/05/14	28
30		Memòria	7 days	Fri 23/05/14	Wed 28/05/14	29
31		Presentació	2 days	Thu 29/05/14	Fri 30/05/14	30
32		Fi de Projecte	0 days	Sat 31/05/14	Sat 31/05/14	31

Figura 2 Detall del pla de treball

2.4.2 Planificació temporal

La planificació temporal segons les tasques definides en l'apartat anterior:

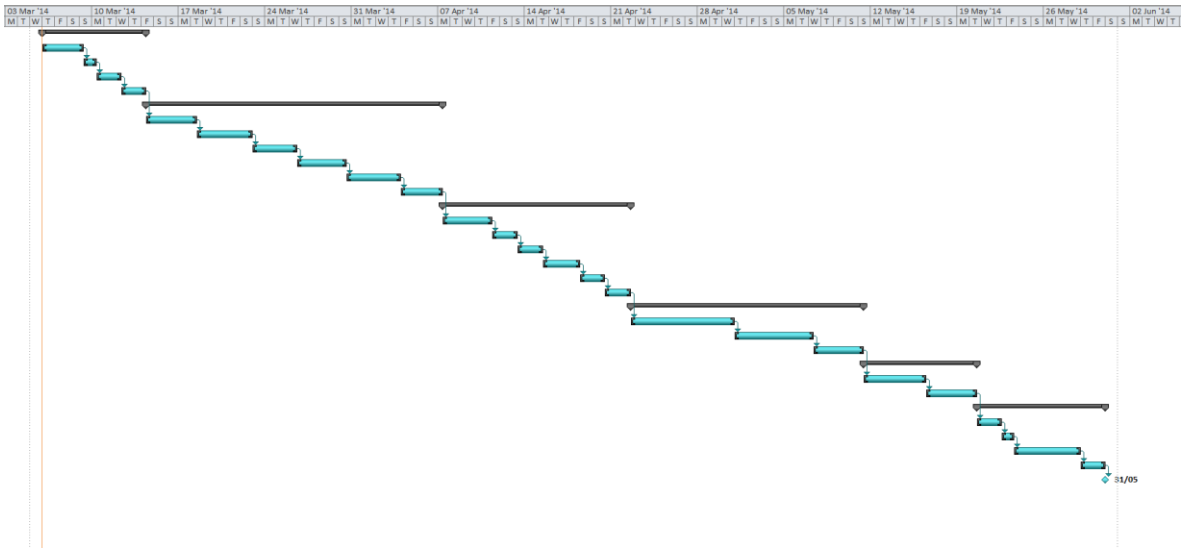


Figura 3 Diagrama de Gantt

Aquesta planificació està subjecte a modificacions durant la vida del projecte degut a la poca previsió dels requeriments i desviacions que poden succeir en treball teòrics sense tenir experiència prèvia.

3 Estudi i anàlisi del problema

En aquest apartat es definirà un exemple sobre el qual el sistema multi-agent podrà treballar. D'aquesta manera es pot aconseguir un prototip molt més detallat al aplicar-lo sobre un exemple concret i es pot obtenir una idea general sobre els reptes amb els que ens podem trobar.

3.1 Un sistema de presa de decisions distribuït

Tal com el títol del projecte indica, el problema a resoldre és la realització d'un sistema de presa de decisions distribuït mitjançant sistemes multi-agent i algorismes classificadors per donar solució a un problema.

En els següents apartats es mostrarà la idea bàsica del problema a resoldre i de com hauria de ser el comportament dels agents i la seva resposta. Per a aquest propòsit, utilitzaré un dels exercicis de la pràctica d'aprenentatge computacional on s'utilitzava un algorisme classificador sobre un conjunt de dades d'entrenament.

3.1.1 Descripció del problema

Per la presentació del problema, utilitzarem un dels exemples de la pràctica d'aprenentatge computacional on es disposava d'un conjunt de dades de 100 casos d'anàlisis de la qualitat de l'esperma. Els resultats dels anàlisis poden ser Normal (N) o Alterat (O). Aquest exemple es pot trobar i descarregar des de el repositori UCI (https://archive.ics.uci.edu/ml/datasets/Fertility).

Per a aquest exemple, utilitzarem 3 agents classificadors. Dels 100 registres que disposem, utilitzarem 95 per entrenar els agents i la resta els repartirem entre aquests en la mateixa proporció, 5 registres per agent per a la seva correcta predicció. Cada agent retornarà una precisió de la classificació obtinguda utilitzant un algorisme d'aprenentatge supervisat. En el nostre cas, utilitzarem un **arbre de decisió** (J48) per la seva familiaritat amb altres assignatures del Grau. Un cop tenim la resposta de cada classificador, aquestes haurien de ser processades per el manager i donar una resposta general a l'usuari.

La següent figura mostra l'experiment a realitzar sobre el problema plantejat:

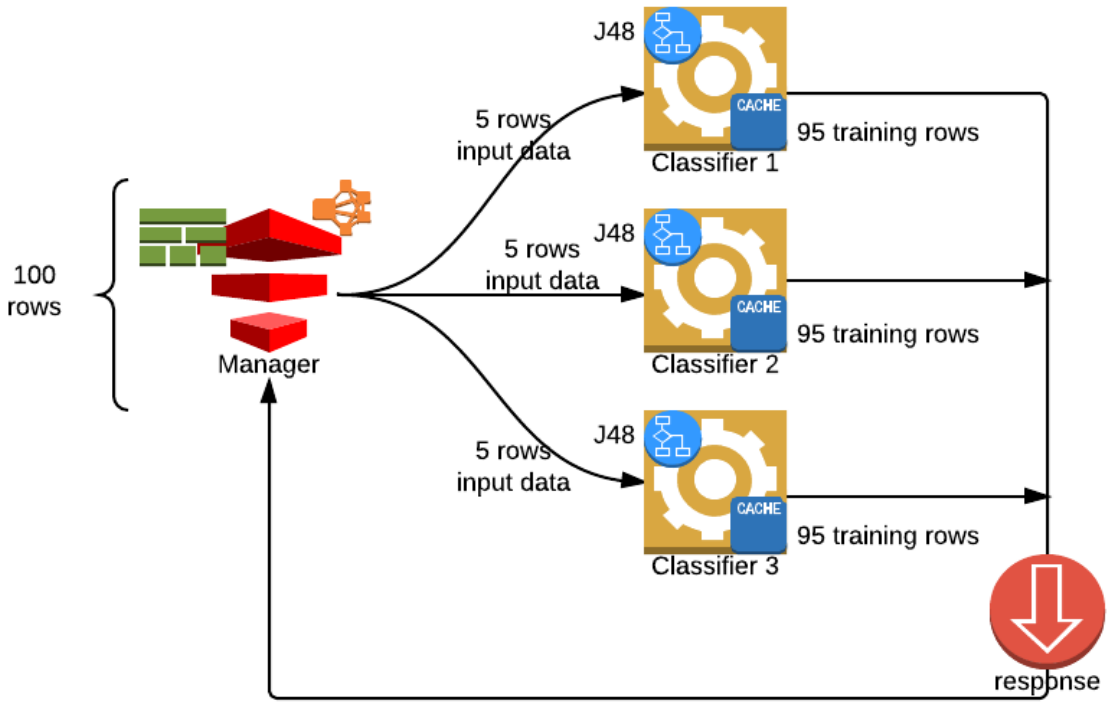


Figura 4 Diagrama distribució de dades (training fixe)

L'execució d'aquesta configuració a través del Weka de manera manual, ens retorna els següents resultats:

Instance to classify	Data	Classified as	Classification by Agent1	Precision Agent 1	Classification by Agent2	Precision Agent 2	Classification by Agent3	Precision Agent 3	Final Decision
1	-1,0.78,1,1,0,1,0.6,-1,0.38	N	N	100% 95/95	N	100% 95/95	N	100% 95/95	N
2	-1,0.78,1,0,1,0,1,-1,0.25	N	N	100% 95/95	N	100% 95/95	N	100% 95/95	N
3	-1,0.56,1,0,1,0,1,-1,0.63	N	N	100% 95/95	N	100% 95/95	N	100% 95/95	N
4	-1,0.67,0,0,1,0,0.6,0,0.5	O	N	100% 95/95	N	100% 95/95	N	100% 95/95	N
5	-1,0.69,1,0,0,0,1,-1,0.31	N	N	100% 95/95	N	100% 95/95	N	100% 95/95	N

Figura 5 Execució de l'algorisme classificador J48 per agent

Els agents han realitzat una classificació i com que han utilitzat el mateix criteri (mateix número de dades per a l'entrenament), la classificació final és de 5 (N) dels 5 registres dels que disposàvem per al seu anàlisi. Com es pot veure, com que cada agent té una precisió del 100% és a dir, utilitza 95 elements com a entrenament, del total de 95 elements disponibles per a l'entrenament de l'agent i per tant a l'hora de prendre la decisió final, només cal comprovar que tots han obtingut el mateix resultat i és el resultat que el manager pren.

En el cas en que cada agent tingués un criteri diferent, caldria considerar un pes diferent per a cada valor i així obtenir un valor en funció del pes de cada agent.

El següent exemple posa de manifest aquest problema i que és la base principal d'aquest projecte. Imaginem doncs que cada agent disposa d'un valor diferent de dades d'entrenament. En aquest cas, sabem que tenim 95 elements disponibles per a entrenar cada agent, però ara, l'agent fa una tria sobre aquests. La següent figura mostra una configuració exemple:

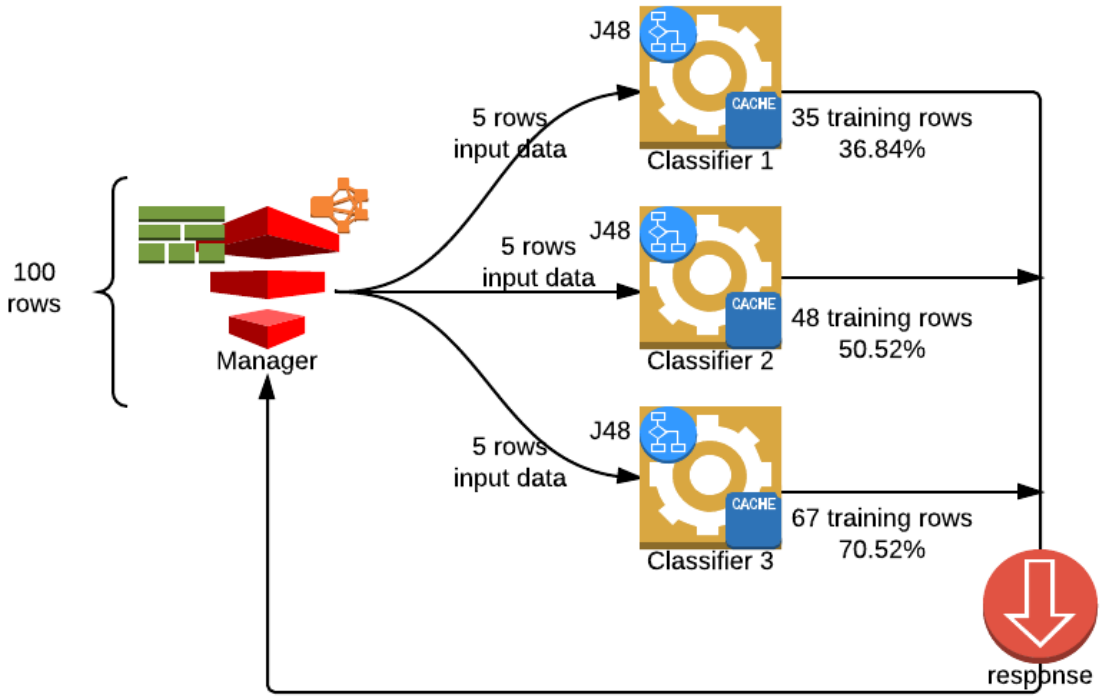


Figura 6 Diagrama de distribució de dades (training variable)

Amb els resultats següents després de l'execució:

Instance to classify	Data	Classified as	Classification by Agent1	Precision Agent 1	Classification by Agent2	Precision Agent 2	Classification by Agent3	Precision Agent 3	Final Decision
1	-1,0.78,1,1,0,1,0.6,-1,0.38	N	N (2)	36.84% 35/95	N (2)	50.52% 48/95	N (2)	70.52% 67/95	N (2)
2	-1,0.78,1,0,1,0,1,-1,0.25	N	O (1)	36.84% 35/95	N (2)	50.52% 48/95	N (2)	70.52% 67/95	N (1.76)
3	-1,0.56,1,0,1,0,1,-1,0.63	N	N (2)	36.84% 35/95	N (2)	50.52% 48/95	N (2)	70.52% 67/95	N (2)
4	-1,0.67,0,0,1,0,0.6,0,0.5	O	O (1)	36.84% 35/95	O (1)	50.52% 48/95	N (2)	70.52% 67/95	O (1.44)
5	-1,0.69,1,0,0,0,1,-1,0.31	N	N (2)	36.84% 35/95	N (2)	50.52% 48/95	N (2)	70.52% 67/95	N (2)

Figura 7 Execució de l'algorisme classificador J48 per agent

Com es pot observar en la taula de la figura 7, l'agent ha pres una decisió final basant-se en els pesos de cada classificador. Per a fer el càlcul d'aquest, s'ha fet servir una simple mitja ponderada i després s'ha considerat el valor final segons la proximitat a un valor numèric donat a cada valor de classificació.

Per tant si donem el valor numèric 1 per a la classificació tipus O i el valor 2 per a la classificació de tipus N, podem utilitzar la mitjana ponderada per a obtenir el valor final per a l'agent classificador.

Així doncs, l'element número 4, després de passar per els 3 classificadors obté un resultat diferent que en el de l'execució de la figura 4. D'aquí obtenim:

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i w_i}{\sum_{i=1}^{n} w_i} = \frac{1 \cdot 36.84\% + 1 \cdot 50.52\% + 2 \cdot 70.52\%}{36.84\% + 50.52\% + 70.52\%} = 1.44$$

Com que O=1 i N=2, podem veure que per a fer una decisió correcta caldria que el valor fos major que >1.5 per a poder definir que el valor ha de ser classificat com a N, per tant el valor cau dins l'àrea per sota del 1.5 i es categoritza com a O.

Per a cada instància s'ha realitzat el mateix càlcul, provocant així la decisió de l'agent sobre certes decisions preses per altres agents.

3.2 Disseny dels agents

Cal que els agents funcionin en un entorn distribuït, és a dir, cal que el manager disposi de la capacitat per a crear un cert número d'agents classificadors a voluntat i que processi tot l'intercanvi de missatges estimat. Aquest sistema multi-agent intel·ligent disposarà de dos tipus d'agent: *manager* i *classificador*. Cada agent haurà d'actuar d'acord amb les seves responsabilitats i negociar els paràmetres que cal que s'enviïn.

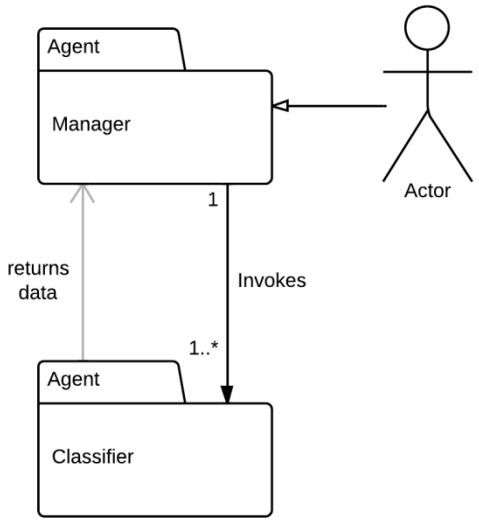


Figura 8 Diagrama de components: Esquema general de funcionament

Tal com es mostra en la figura 6, la solució està composada a grans trets per dues entitats principals:

- **Agent Manager**: S'encarregarà de carregar les dades del repositori i actualitzar i crear els agents classificadors d'acord amb la configuració de l'usuari. Restarà a l'espera de la resposta d'aquests i retornarà les dades a l'usuari d'acord a uns criteris definits.
    - **Com a propietats principals, aquest defineix**:
        - Localització del repositori des d'on obtenir les dades (Fitxer ARFF).

          Exemple de configuració:
            - `ArffDataSetLocation=fertality_Diagnosis_train.arff`
        - Percentatge a usar com a entrenament sobre les dades originals. Valor per defecte = ***70%*** de la mida del fitxer.

Exemple de configuració:

- `PercentageTrainingData=70`

- Número de classificadors a crear. Valor per defecte ***3***. Aquest valor serveix per a indicar a l'agent, que ha de crear 3 agents per defecte. Aquesta propietat està connectada amb la propietat següent (Mòdul classificador a utilitzar) i que caldrà crear en funció del número d'agents que tinguem definits en aquesta propietat.

  Exemple de configuració:

  - `NumberOfClassifiers=3`

- Mòdul classificador a utilitzar. Valor per defecte: ***random***. Aquesta propietat és dinàmica i cal crear tantes variables com agents classificadors tinguem. Els valors que accepta aquesta propietat son:

  - **J48** (Utilitza un arbre classificador)
  - **IBk** (Utilitza algorisme basat en el veí més proper)
  - **MLP** (Utilitza una xarxa neuronal)
  - **random** (Deixa la decisió per al node classificador)

  Exemple de configuració si disposem de **3 classificadors**:

  - `ClassifierModule1=J48`
  - `ClassifierModule2=IBk`
  - `ClassifierModule3=MLP`

- Classificadors utilitzen el mateix contenidor. Aquesta propietat defineix si la creació de nous nodes es realitza en el mateix contenidor o no. El node manager apareix sota el "Main-Container" i la resta de nodes es creen en diferents containers o en un d'agrupat. D'aquesta manera obtenim la configuració mencionada per la figura 1 on el node classificador està ubicat dins el mateix contenidor. Valor per defecte ***fals***. Veure apartat del Jade per a entendre el significat d'aquesta propietat.

Exemple de configuració:

- `ClassifiersUseSameContainer=false`

- Fitxer de sortida. Aquesta propietat indica on es desarà el fitxer generat per el sistema amb la informació que cada classificador aporta al manager.

  Exemple de configuració:

  - `OutputFileLocation= fertality_Diagnosis_train.out`

- Verbositat: Aquesta propietat defineix si es vol mostrar la informació que els agents estan tractant per pantalla. Valor per defecte: ***false***.

  Exemple de configuració:

  - `Verbosity=false`

- Logging: Aquesta propietat defineix si es vol desar la informació que cada agent mostra per pantalla utilitzant log4j. Valor per defecte: ***false***.

  Exemple de configuració:

  - `Logging=false`

- **Agent Classificador**: S'encarregarà de classificar les dades enviades pel Manager d'acord amb el classificador escollit. Aquest agent disposarà d'una *Factory* de classificadors que el permetrà utilitzar el classificador que l'usuari hagi definit.
  - **Com a propietats principals, aquest defineix**:
    - Ruta de l'aplicació: Ruta des d'on l'aplicació (clients) s'executen. Aquest valor és el que es transfereix a l'agent i que s'utilitza per la construcció del repositori weka individual per a cada agent.

      Exemple de configuració:

      - `ApplicationPath=workspace/ddm/source/`

- Valor mínim a considerar per al valor de percentatge d'entrenament. Per defecte 30%. Aquest valor indica el percentatge mínim de dades que el classificador utilitzarà per a entrenar, és a dir, del conjunt de dades, mai triarà un valor inferior al 30% d'aquests.

  Exemple de configuració:

  - `MinimumPercentageTraining=30`

- Valor màxim a considerar per al valor de percentatge d'entrenament. Per defecte 100%. Aquest valor indica el percentatge màxim de dades que el classificador utilitzarà per a entrenar, és a dir, del conjunt de dades, mai triarà un valor superior al 95% d'aquests.

  Exemple de configuració:

  - `MaximumPercentageTraining=95`

- Verbositat: Aquesta propietat defineix si es vol mostrar la informació que els agents estan tractant per pantalla. Valor per defecte: ***false***.

  Exemple de configuració:

  - `Verbosity=false`

- Logging: Aquesta propietat defineix si es vol desar la informació que cada agent mostra per pantalla utilitzant log4j. Valor per defecte: ***false***.

  Exemple de configuració:

  - `Logging=false`

3.2.1 Agent Manager

La següent figura mostra l'estructura bàsica de l'agent Manager i les diferents classes que fa servir per a la seva correcta execució.

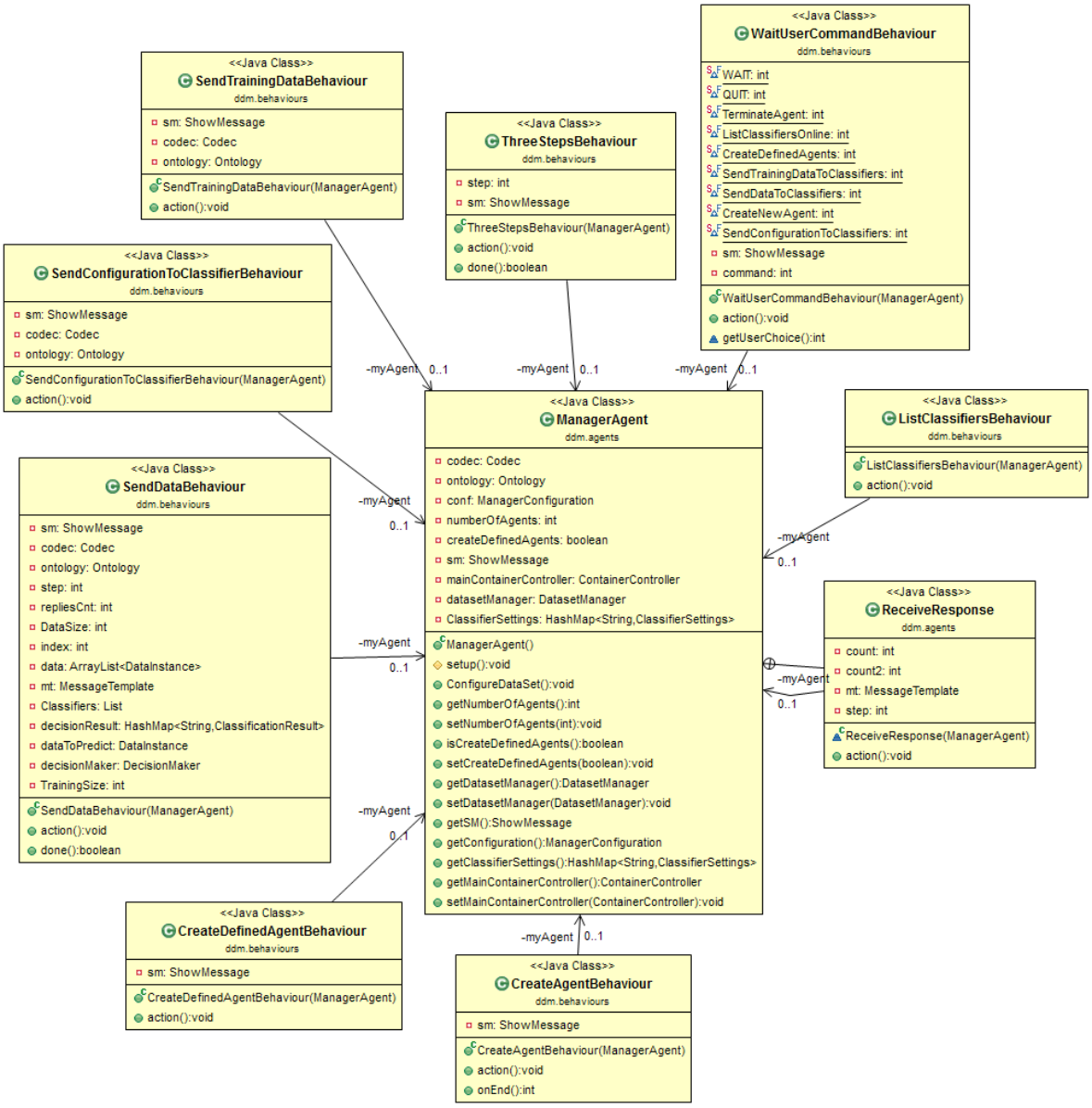


Figura 9 Diagrama de classes Agent Manager

El Manager disposa d'uns paràmetres basics tals com el Id d'agent, el llenguatge i la ontologia a usar. A més a més disposa d'una sèrie de ***behaviours*** que el permetran interaccionar amb les dades que el client envia per a fer la presa de decisió final.

3.2.2 Agent Classificador

El següent esquema (figura 10) mostra un conjunt de classificadors o cadascun calcula la seva conclusió per a una determinada entrada x. L'agent manager disposa d'un mòdul "*combine*" que realitzarà la combinació de les conclusions de cada classificador i prendrà una decisió final $M_{(x)}$.

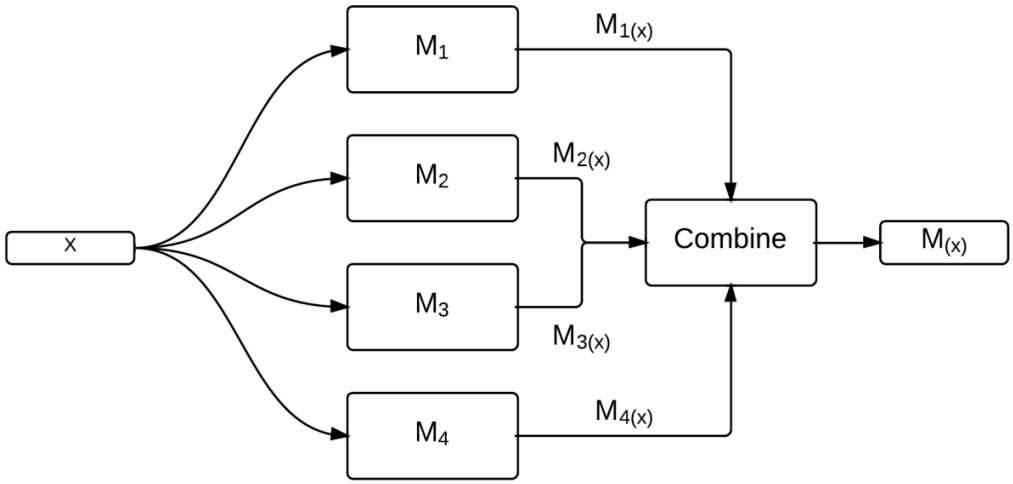


Figura 10 Esquema conjunt de classificadors

L'agent classificador disposa de les mateixes propietats bàsiques de qualsevol agent: Identificador, llenguatge i ontologia. A més disposarà d'una sèrie de *behaviours* que el permetran interaccionar amb les dades rebudes per el Manager i així poder realitzar les operacions determinades per al classificador.

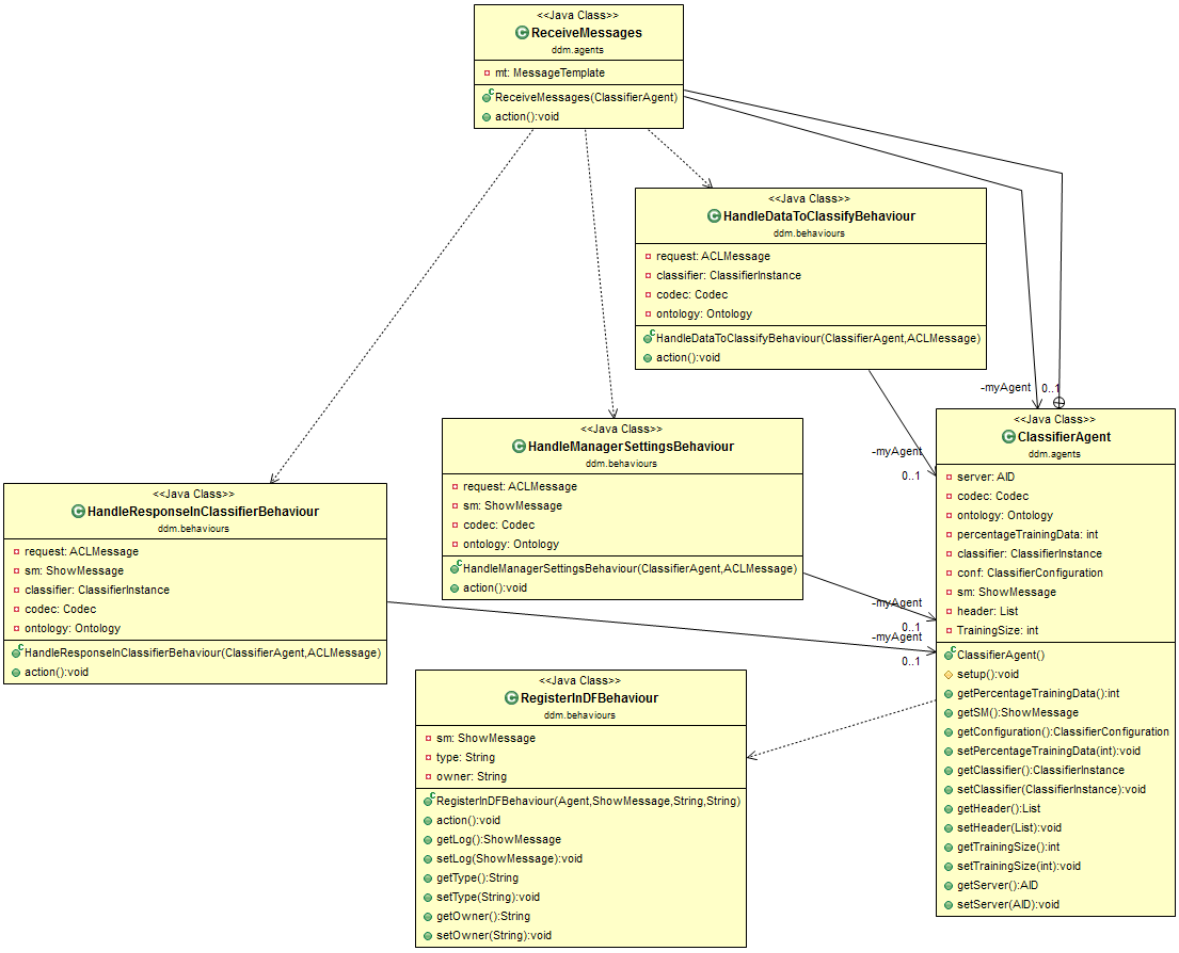


Figura 11 Diagrama de classes Agent Classificador

L'agent classificador és molt més senzill ja que només ha de realitzar un seguit d'operacions en funció de les dades rebudes. Cada classificador genera una factoria de classificadors i en funció d'uns paràmetres de configuració, escollirà un tipus o un altre per a la funció de classificació.

3.2.2.1 Factoria de classificadors

Cada agent disposa d'una factoria de classificadors. Per a la realització d'aquest projecte, s'han implementat els següents classificadors:

- Arbre de decisió (J48)
- Veí més proper (IBk)
- Xarxa neuronal (MLP)

Una factoria és un patró molt comú per crear objectes sense especificar-ne la classe exacta utilitzant un mètode de creació. En aquest cas, cada agent disposa d'una instància de la factoria i utilitzarà un dels classificadors a petició.
Un altra avantatge de la utilització d'aquesta arquitectura és que podem estendre la llista de classificadors sense alterar els agents i així aconseguir un sistema molt més flexible i amb més habilitats.

En la següent figura podem veure el disseny de la factoria:

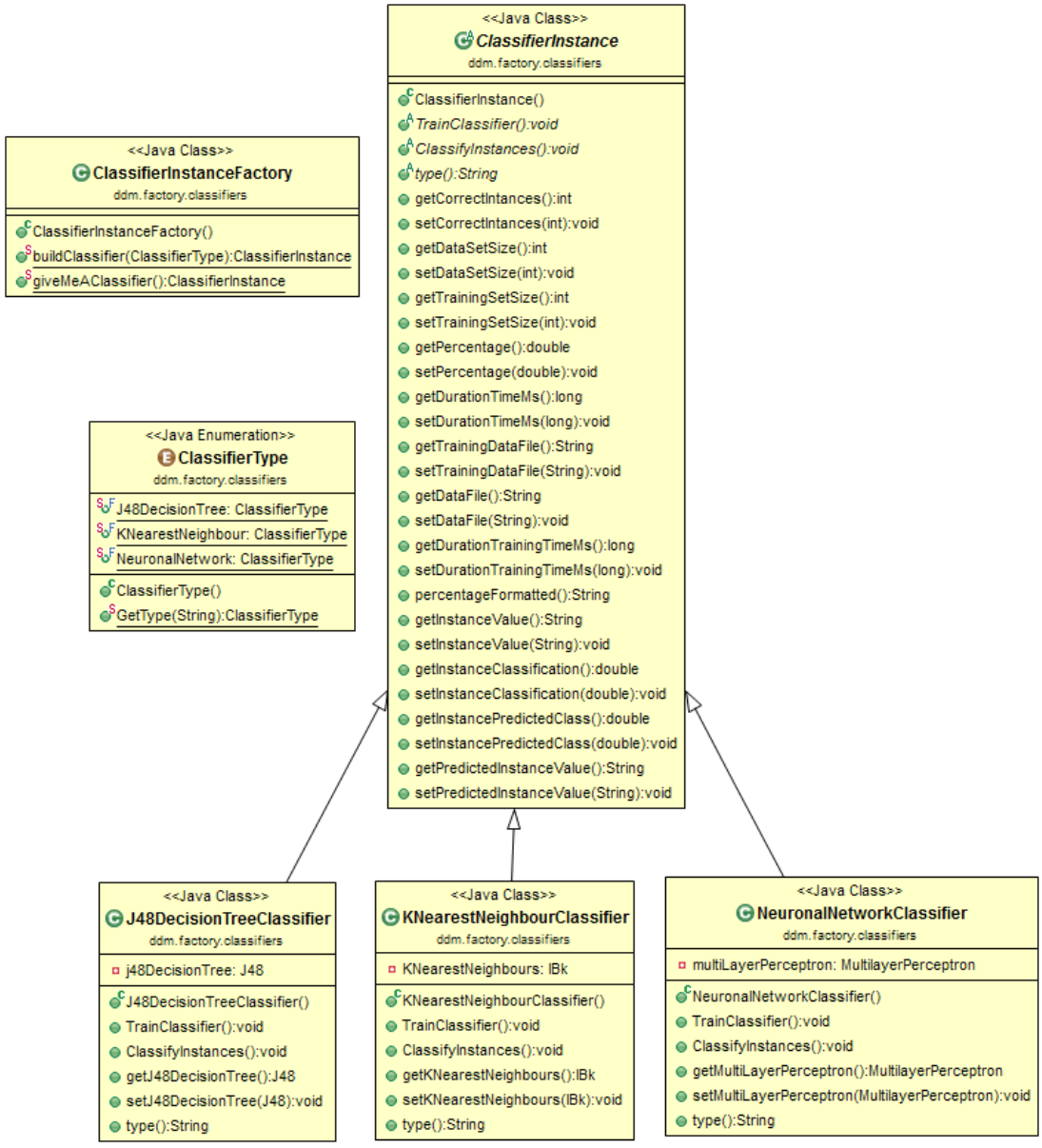


Figura 12 Factoria classificadors

Cada agent crea una instància de classificació durant la seva creació. Inicialment aquest classificador és aleatori ja que és tasca del classificador obtenir-ne un. Durant la conversa inicial del Manager cap als agents, es defineix la configuració d'aquests classificadors i a partir d'aquí el manager pot entrenar cada classificador i enviar les dades a classificar.

La definició del classificador és molt senzilla i només cal fer la següent crida per a que el classificador aconsegueixi una instància del weka que podrà usar durant la seva execució:

```
ClassifierInstance classifier = ClassifierInstanceFactory.giveMeAClassifier();
```

El mètode giveMeAClassifier() fa servir una funció aleatòria per escollir un dels classificadors disponibles en la factoria:

```
	public static ClassifierInstance giveMeAClassifier(){
		Random randomGenerator = new Random();
		int randomInt = randomGenerator.nextInt(2);
		ClassifierInstance classifier = null;
		switch (randomInt) {
		case 0:
			classifier = new J48DecisionTreeClassifier();
			break;
		case 1:
			classifier = new KNearestNeighbourClassifier();
			break;
		case 2:
			classifier = new NeuronalNetworkClassifier();
			break;
		}
		return classifier;
	}
```

Com es pot veure en la figura anterior, cada classificador d'instàncies disposa de dos mètodes generals, un per a entrenar el classificador i un altre per a classificar una instància segons els paràmetres que s'han definit un cop s'ha entrenat l'agent.

Cal considerar doncs, que es pot entrenar tantes vegades com vulguem l'agent ja que és una tasca totalment independent. Ara bé, el sistema defineix que s'han d'entrenar tots els agents a l'hora. Es podria arribar a configurar el sistema per acceptar configuracions individuals, però restaria com a millora del projecte.

Com podem veure en els següent diagrama de seqüència, es pot veure com funciona la creació bàsica dels agents i com es configura cada classificador:

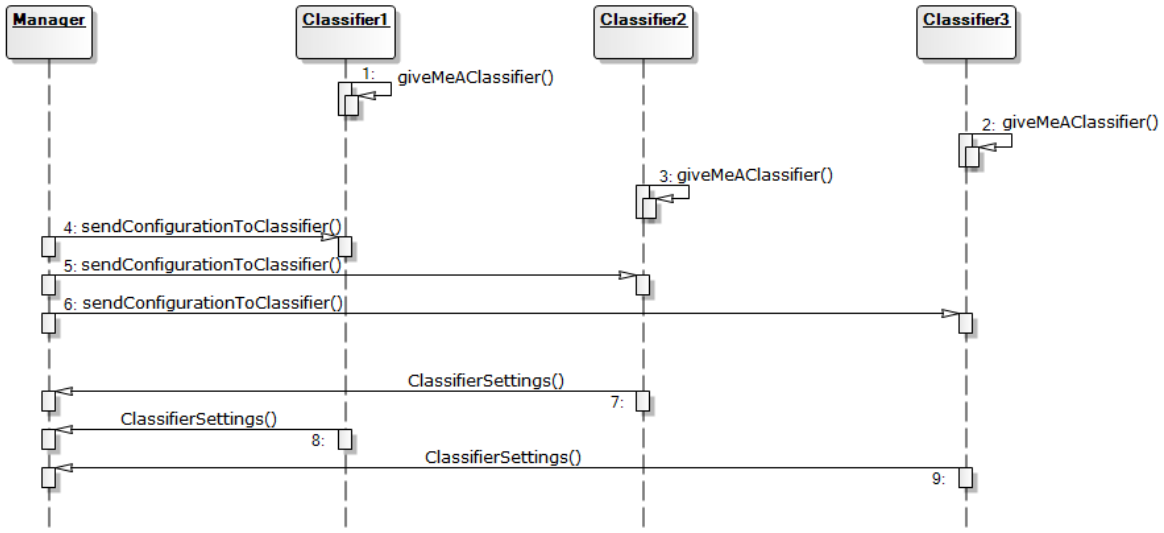


Figura 13 Configuració instàncies classificadores

Com es pot veure en el diagrama de seqüència, cada classificador fa una elecció inicial del tipus de classificador i manté aquesta instància durant la vida del classificador. El manager és capaç d'enviar una configuració addicional que configura a través del seu fitxer de configuració. Un cop el manager ha aconseguit enviar la informació, cada classificador envia una resposta amb les dades canviades per a que el manager pugui consolidar aquestes i pugui realitzar els canvis pertinents a les variables que conté amb informació de cada agent per a després realitzar els càlculs adients per a la decisió final.

A part del tipus de classificador, cada Classificador obté un valor numèric aleatori per a les propietats: Valor mínim a considerar per al valor de percentatge d'entrenament i Valor màxim a considerar per al valor de percentatge d'entrenament. Aquests dos paràmetres s'envien al Manager per tal de poder enllestir la configuració del repositori a enviar.

La configuració d'aquest valor es fa a través de:

```
this.percentageTrainingData = DataUtils.RandomNumber(conf.getMinimumPercentageTraining(),
conf.getMaximumPercentageTraining());
```

3.2.3 Conversació entre agents

Un cop els agents han sigut creats i correctament configurats, des de la consola de comandes podem inicial la conversació entre agents que ens permetrà executar

l'algorisme principal que proposa aquest projecte i donar així una decisió final sobre un sistema de presa de decisions distribuït.

En el següent diagrama de seqüència es pot veure una seqüència bàsica de comunicació entre agents:

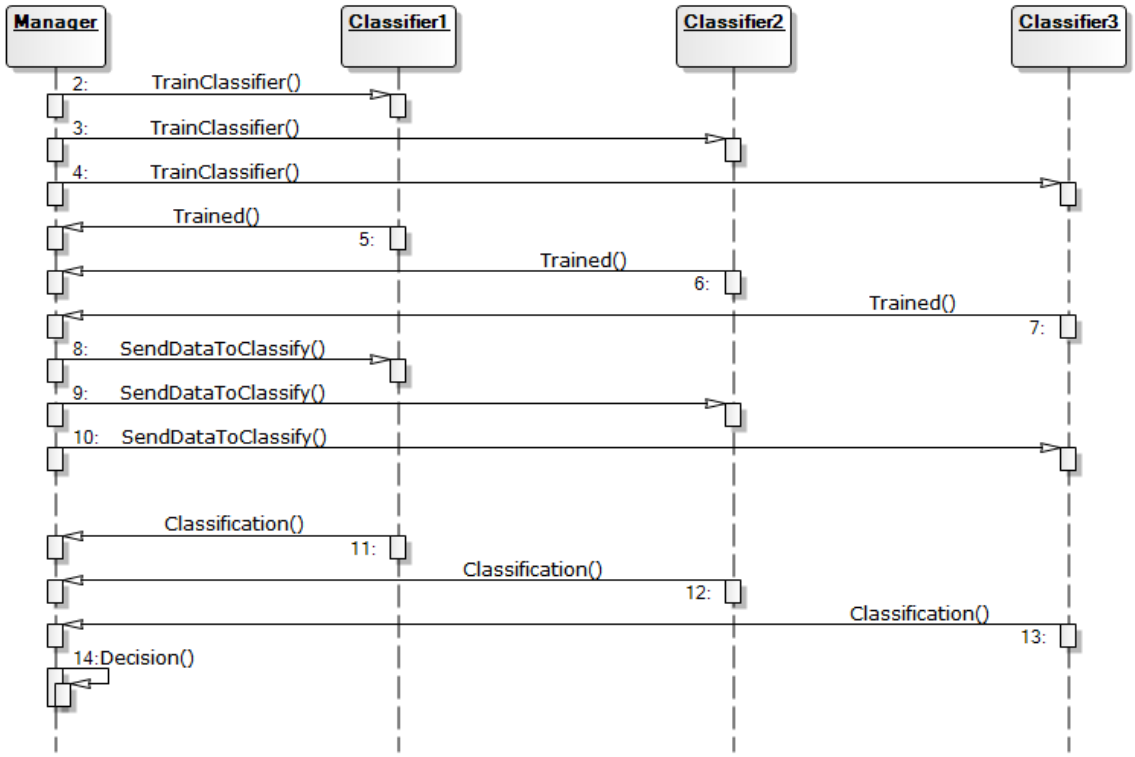


Figura 14 Comunicació entre agents

Com hem vist anteriorment, durant la primera fase de la comunicació, els agents parlen entre ells (Manager - Classificador) i d'aquesta manera es crea una comunicació on s'entenen ja que d'aquesta transició el que es vol obtenir és una configuració bàsica de cada agent per tal de crear les dades necessàries per a cada agent.

Un cop les dades son preparades, el sistema està preparat per a enviar les dades d'entrenament. Un cop cada classificador ha sigut entrenat, aquest envia la resposta al Manager. El manager acumula les respostes per així saber que tots els classificadors estan llestos per a treballar.

A partir d'aquí, el Manager té el conjunt de dades a processar i a enviar a cada agent per a la seva tasca de classificació. El protocol de comunicació és molt bàsic i l'agent manager simplement genera un *behaviour* especial que s'encarrega de segregar les dades del repositori i envia cada dada individual a cada agent classificador mitjançant el mètode *SendDataToClassify*(). Llavors un cop enviades les dades, queda a l'espera de la resposta de la classificació i un cop rep totes les respostes pren una decisió final i la desa en un fitxer amb les dades originals per al seu posterior anàlisi.

Un cop ha realitzat aquests passos, torna al seu estat inicial on continuarà enviant dades fins a que s'hagin acabat aquestes del repositori.

4 Implementació del sistema multiagent amb JADE

4.1 Sistema de configuració i inicialització

El sistema funciona de manera semi automàtica. Durant l'inici de l'aplicació, cal executar una comanda per instal·lar l'agent manager al servei del Jade i un cop aquest està en línia, es poden executar les comandes necessàries al mateix agent que és el que crearà els diferents classificadors en funció de la configuració definida. A més a més es podran anar afegint agents al vol, és a dir, en qualsevol moment, podrem anar creant agents i fer que aquests interaccionin de la mateixa manera que els agents classificadors predefinits pels fitxers de configuració.

Per defecte l'agent manager s'instancia en un contenidor principal i la resta de nodes es distribueixen en contenidors individuals que es poden configurar. Les propietats per defecte son les següents:

```
Runtime rt = Runtime.instance();
Profile p = new ProfileImpl();
p.setParameter(Profile.MAIN_HOST, "localhost");
p.setParameter(Profile.MAIN_PORT, "1099");
p.setParameter(Profile.CONTAINER_NAME, "Main-Container");
```

On cada agent classificador utilitza la màquina des d'on s'executa l'agent manager i amb el port 1099.

Es podria millorar aquesta secció afegint una sèrie de paràmetres per a la correcta definició remota però també es pot fer des de la interfície d'usuari del Jade.

Per a la creació de cada agent, es fa servir un *behaviour* que és capaç de crear un **AgentController** i iniciar-lo mitjançant la comanda *start()*. A més a més, cada vegada que s'inicia un agent, aquest quedarà a l'espera durant uns 2 segons per a que li doni temps a iniciar tots els protocols interns ja que el manager voldrà comunicar-hi de seguida per saber si es troba en línia o no.

```
public class CreateAgentBehaviour extends OneShotBehaviour {

	private ShowMessage sm;
	private ManagerAgent myAgent;

	public CreateAgentBehaviour(ManagerAgent a) {
		super(a);
		myAgent = a;
		this.sm = a.getSM();
	}

	@Override
	public void action() {
		sm.Log("Create Agent - start");
		Runtime rt = Runtime.instance();
		Profile p = new ProfileImpl();
		p.setParameter(Profile.MAIN_HOST, "localhost");
		p.setParameter(Profile.MAIN_PORT, "1099");
		p.setParameter(Profile.CONTAINER_NAME, "Main-Container");

		ContainerController cc = null;
		if (myAgent.getConfiguration().getClassifiersUseSameContainer()) {
			if (myAgent.getContainerController() == null)
				myAgent.setMainContainerController(rt.createAgentContainer(p));
			cc = myAgent.getContainerController();

		} else
			cc = rt.createAgentContainer(p);
		if (cc != null) {
			try {
				int agentId = myAgent.getNumberOfAgents() + 1;
				AgentController ac = cc.createNewAgent("Classifier" + agentId,
"ddm.agents.ClassifierAgent", null);
				ac.start();
				myAgent.setNumberOfAgents(myAgent.getNumberOfAgents()+1);
			} catch (Exception e) {
				e.printStackTrace();
			}
		}
		sm.Log("Create Agent - end");
	}

	@Override
	public int onEnd(){
		try {
			Thread.sleep(2000);
		} catch (InterruptedException e) {
			e.printStackTrace();
		}
		return super.onEnd();
	}
}
```

Es manté un comptador intern que indica el número de classificadors creats. D'aquesta manera es poden anar creant tants com vulguem i amb un índex.

4.2 Arquitectura JADE

L'objectiu d'aquest projecte és el de realitzar un sistema de presa de decisions distribuït amb l'arquitectura Jade i l'eina de mineria Weka. Cal doncs, presentar l'arquitectura JADE i veure a grans trets quin és el seu funcionament bàsic per tal d'entendre la complexitat de la solució proposada.

JADE (Java Agent Framework de Desenvolupament) és un framework de programari totalment implementat en el llenguatge Java. Simplifica la implementació de sistemes multi-agent a través d'un middle-ware que compleixi amb les especificacions FIPA i a través d'un conjunt d'eines gràfiques que suporta les fases de depuració i implementació. La plataforma d'agents pot ser distribuïda a en diferents màquines i la configuració es pot controlar a través d'una interfície gràfica remota. La configuració es pot canviar en temps d'execució i moure agents d'una màquina a una altra, segons les necessitats de l'aplicació.[5]

Els paquets principals de l'arquitectura son:

Jade.core	Implementa el kernel del Jade, entorn run-time distribuït que suporta la plataforma i les seves eines.
Jade.content	Conté una col·lecció de classes que proporciona als programadors les eines necessàries per manipular complexes expressions segons un llenguatge i ontologia.
Jade.domain	Conté la implementació dels agents AMS i DF que son especificats per els estàndards de la FIPA.
Jade.gui	Conté alguns components Java que poden ser útils per a construir interfícies gràfiques.
Jade.imtp	Conté el JADE IMTP (Internal Message Transport Protocol) que està basat en RMI.
Jade.lang.acl	Conté el suport de FIPA Agent Communication Language (ACL).
Jade.mtp	Conté interfícies que han de ser implementades per JADE MTP.

L'arquitectura executa els diferents agents en contenidors que s'executen en una plataforma. Aquests es poden definir com a dominis d'agent i es poden distribuir en diferents equips i xarxes.

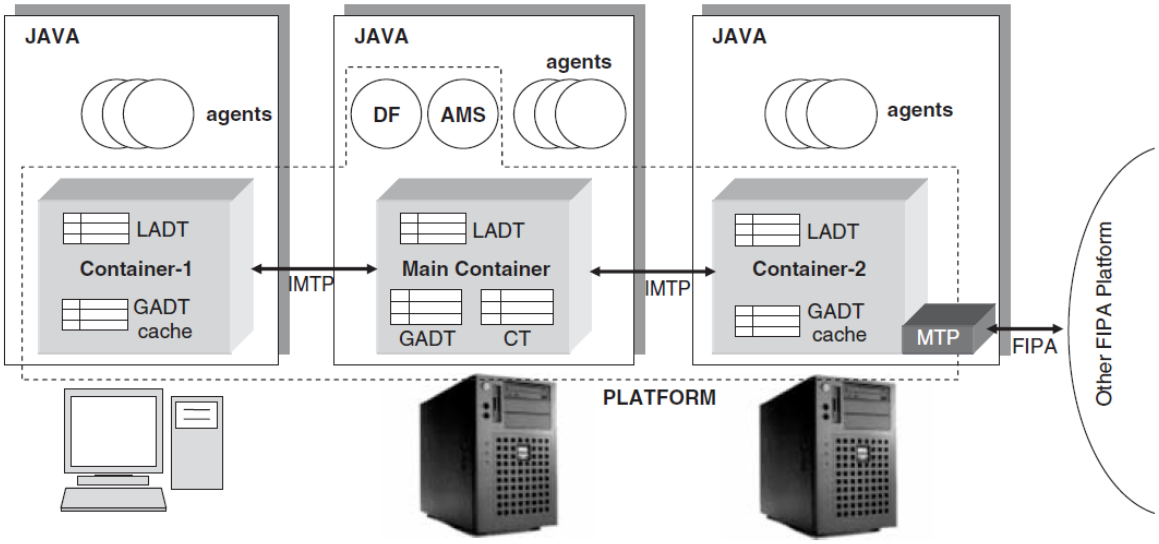


Figura 15 Relació entre els components de l'arquitectura principal del JADE[6]

Com es pot veure en la següent figura, el projecte es configura de manera que cada agent classificador pot utilitzar un contenidor diferent ja que el JADE ens permet realitzar aquesta configuració:

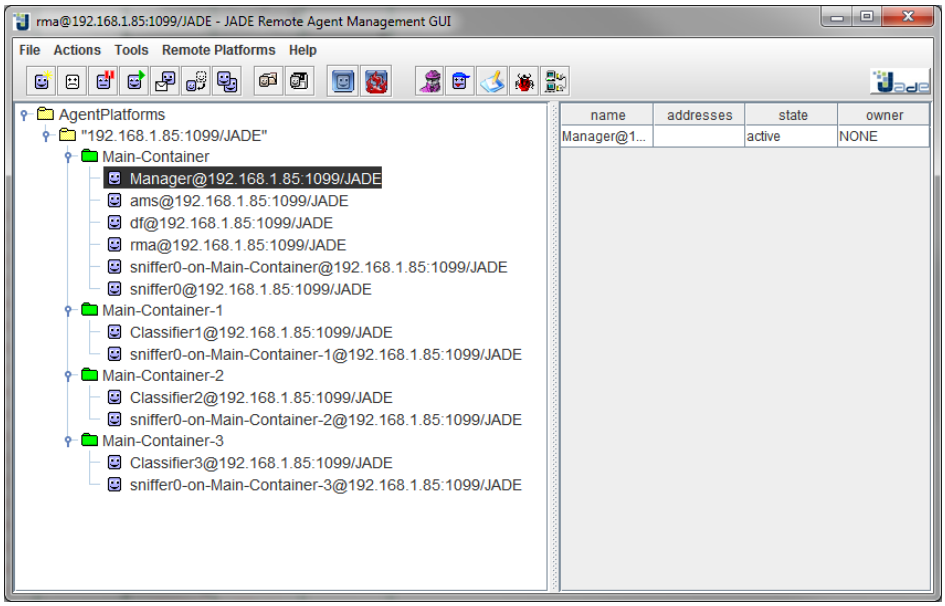


Figura 16 Contenidors del JADE

Mitjançant l'opció de configuració del manager (*ClassifiersUseSameContainer*), podem definir si volem que tots els classificadors estiguin sota el mateix contenidor o si volem contenidors individuals per a cada agent. Mitjançant aquests paràmetres podem definir els agents distribuïts i com volem aquesta distribució.

4.3 Behaviours dels agents

Els agents realitzen tasques mitjançant behaviours. Cada **behaviour** ha d'estendre la classe **jade.core.bahaviours.Behaviour** i per a que cada **behaviour** sigui executat cal que s'afegeixi a l'agent mitjançant la crida **addBehaviour()**. Els Behaviours poden ser afegits a qualsevol time, per exemple quan l'agent s'inicia (mètode **setup()**) o dins d'altres behaviours.

Cada classe que estén la classe behaviour ha d'implementar dos mètodes abstractes. L'acció **action()** que defineix les operacions que es realitzaran i el mètode **done()** que retorna un valor booleà per indicar si un behaviour s'ha completat o no per així ser remogut de la llista de behaviours que l'agent està executant.

Cal entendre bé com funcionen els behaviours ja que un agent pot executar diferents behaviours simultàniament. Aquests executen el mètode **action()** i s'esperen fins a que acabin l'execució.

A l'hora de programar-los cal tenir en compte que s'executa dins el mètode **action()** i com behaviours poden interaccionar amb altres behaviours i produir comportaments erronis o estranys. El següent diagrama d'execució mostra el camí que executa el fil de l'agent per a cada behaviour. Cada agent utilitza un fil d'execució (single thread) i es fa per a millorar-ne el rendiment i millorar la sincronització entre behaviours.

El fil d'execució obté la llista de behaviours i els executa un a un amb l'acció **action()** fins que el mètode **done()** s'ha validat.

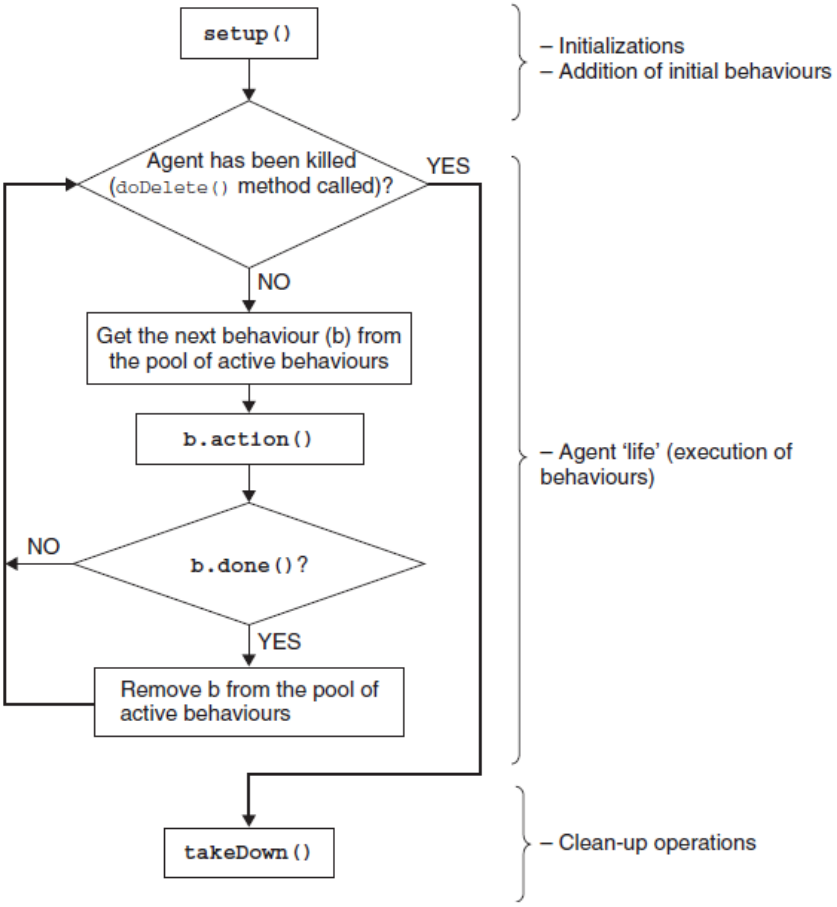


Figura 17 Camí d'execució del fil de l'agent

En el present projecte s'han fet servir behaviours del tipus:

- **One-shot behaviour**: Enviament de dades del Manager a classificador.
- **Cyclic Behaviour**: Processament de respostes dels agents.
- **Generic behaviour**: Sistemes complexes de comunicació amb múltiples enviament i respostes.

El següent exemple mostra un sistema complex de comunicació on el node manager envia 3 paquets (1 per cada agent classificador) i llavors queda a l'espera de les 3 respostes per continuar enviant dades fins que ja no en queden més:

```
public class SendDataBehaviour extends Behaviour {

        private ManagerAgent myAgent;
        private ShowMessage sm;
        private Codec codec = new SLCodec();
        private Ontology ontology = ClassifierOntology.getInstance();
        private int step = 0;
        private int repliesCnt = 0;
```

```
        private int DataSize = 0;
        private int index = 0;
        private ArrayList<DataInstance> data;
        private MessageTemplate mt = MessageTemplate.MatchPerformative(ACLMessage.PROPOSE);
        private jade.util.leap.List Classifiers;
        private HashMap<String, ClassificationResult> decisionResult = new HashMap<String,
ClassificationResult>();
        private DataInstance dataToPredict;
        private DecisionMaker decisionMaker;
        private int TrainingSize;

        // Get the user command
        public SendDataBehaviour(ManagerAgent a) {
                super(a);
                myAgent = a;
                sm = a.getSM();
        }

        public void action() {
                sm.Log("SendDataBehaviour - start");
                switch (step) {
                case 0:
                        //Prepare the data
                        DatasetManager datasetManager = myAgent.getDatasetManager();
                        this.data = datasetManager.getDatasetsData();
                        this.DataSize = data.size();
                        this.TrainingSize = datasetManager.getPartitionList().getTrainingSize();
                        decisionMaker = new DecisionMaker(myAgent.getConfiguration());
                        Classifiers = JadeAgents.SearchAgents(myAgent, "Classifier Agent", null);
                        sm.Log("Agents found: " + Classifiers.size());
                        System.out.println("Data size" + this.DataSize);
                        System.out.println("GOING TO STEP 1");
                        step = 1;
                        break;
                case 1:
                        //Send the data to classify to every agent
                        repliesCnt = 0;
                        this.dataToPredict = null;
                        decisionResult.clear();
                        sm.Log("Looking for agents online");

                        Iterator<?> classifiers = Classifiers.iterator();

                        ACLMessage msg = new ACLMessage(ACLMessage.REQUEST);
                        while (classifiers.hasNext()) {
                                AID server = (AID) classifiers.next();
                                System.out.println("Sending to: " + server.getName());
                                msg.setLanguage(codec.getName());
                                msg.setOntology(ontology.getName());
                                try {
                                        DataInstance dataInstance = data.get(index);
                                        this.dataToPredict = dataInstance;
                                        System.out.println("Data to send " +
dataInstance.toString());
                                        myAgent.getContentManager().fillContent(msg, new
Action(server, dataInstance));
                                        msg.addReceiver(server);
                                        myAgent.send(msg);
                                        try {
                                                Thread.sleep(100);
                                        } catch (InterruptedException e) {
                                                e.printStackTrace();
                                        }
                                        //System.out.println("Contacting client... Please wait!");
                                } catch (Exception ex) {
                                        ex.printStackTrace();
                                }
                        }
```

```
                        if (index < DataSize)
                        {
                                index++;
                                System.out.println("GOING TO STEP 2");
                                step = 2;
                        }
                        break;
                case 2:
                        //Receive all the proposals from every agent
                        ACLMessage reply = myAgent.receive(mt);
                        if (reply != null) {
                                if (reply.getPerformative() == ACLMessage.PROPOSE) {
                                        //System.out.println("Propose!!!");
                                        ContentElement content = null;
                                        try {
                                                content =
myAgent.getContentManager().extractContent(reply);
                                        } catch (Exception ex) {
                                                ex.printStackTrace();
                                        }
                                        Concept action = ((Action) content).getAction();
                                        if (action instanceof ClassificationResult) {
                                                ClassificationResult cr = (ClassificationResult)
action;
                                                System.out.println("*****" + cr.getName() + " " +
cr.getType() + " " + cr.getDuration() + "ms NumCorrect: "
                                                + cr.getNumCorrect() + " Value:" +
cr.getInstanceValue() + " TrainingSize:" + cr.getTrainingSize()
                                                + " Total:" + this.TrainingSize + " Val:" +
cr.getInstanceClassification()
                                                + " Pred:" + cr.getInstancePredictedValue() + "
Value:" + cr.getPredictedInstanceValue());
                                                repliesCnt++;
                                                decisionResult.put(cr.getName(), cr);
                                                if (repliesCnt >= (4*myAgent.getNumberOfAgents()) &&
(index < DataSize))
                                                {
                                                        decisionMaker.Make(dataToPredict,
decisionResult, TrainingSize);
                                                        step = 1;
                                                        try {
                                                                Thread.sleep(1000);
                                                        } catch (InterruptedException e) {
                                                                e.printStackTrace();
                                                        }
                                                        System.out.println("GOING TO STEP 1");
                                                }
                                                else if (repliesCnt >=
(4*myAgent.getNumberOfAgents()) && (index == DataSize))
                                                {
                                                        System.out.println("GOING TO STEP 3");
                                                        try {
                                                                Thread.sleep(1000);
                                                        } catch (InterruptedException e) {
                                                                e.printStackTrace();
                                                        }

                                                        step = 3;
                                                }
                                        }
                                }
                        }
                        else
                        {
                                block();
                        }
                        break;
                case 3:
                        decisionMaker.CloseFile();
```

```
                        myAgent.addBehaviour(new WaitUserCommandBehaviour(myAgent));
                        step = 4;
                        break;
                }
                sm.Log("SendDataBehaviour - end");
        }

        @Override
        public boolean done() {
                return step == 4;
        }

}
```

Per a la comunicació de les dades a classificar, ha calgut realitzar un behaviour una mica més complex ja que calia enviar un missatge idèntic a tots els classificadors i llavors quedar a l'espera de tots els missatges i prendre una decisió amb aquestes respostes.

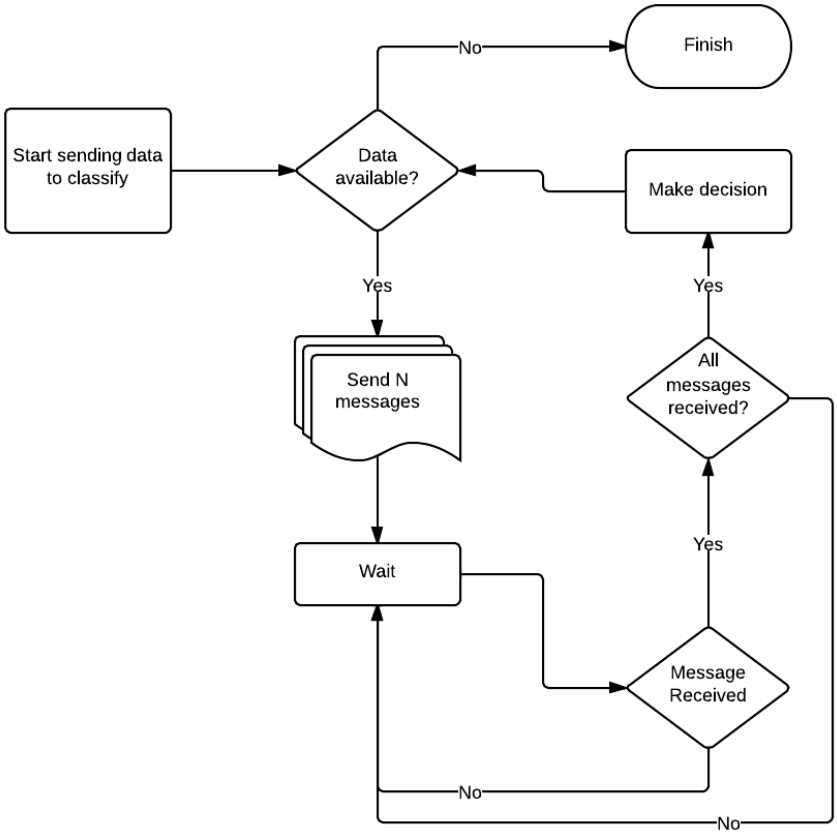


Figura 18 Exemple processament missatges a classificar

En la figura, podem veure com es realitza la lògica d'enviament de missatges i com el sistema queda a l'espera fins que tots els missatges s'han rebut. A partir d'aquí es realitza la tasca de decisió i es continuen processant les dades d'una en una.

4.4 Sistema de comunicació

Una de les característiques més importants dels agents JADE es que tenen l'habilitat de comunicar. El paradigma de comunicació adoptat és el de pas de missatges asíncron. Cada agent té un tipus de bústia (cua de missatges) on JADE desa els missatges enviats per altres agents. En el moment que un missatge és desat en la cua de l'agent, aquest és notificat i quan l'agent agafa aquest missatge, és tasca del programador processar-lo i realitzar després les accions necessàries (informar altres agents o quedar a l'espera, etc.).

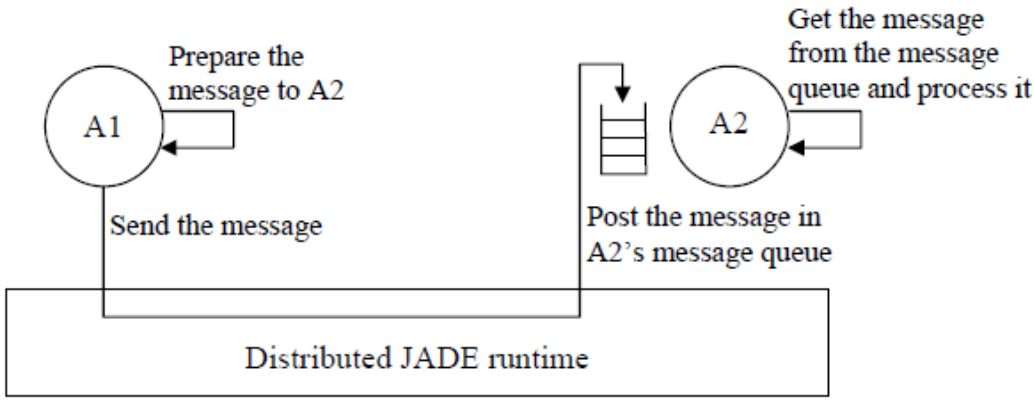


Figura 19 Paradigma de l'enviament de missatges asíncron amb JADE

Cal entendre molt bé aquest funcionament ja que tots els *behaviours* definits s'atendran a aquest tipus de comunicació i haurem de saber com processar les dades que estan a la cua de cada agent.

4.4.1 Paquets de comunicació

JADE està implementat completament sobre les especificacions FIPA i a més a més estén aquest model en aspectes com la interoperabilitat i altres àrees posant de manifest que l'arquitectura JADE és totalment compatible amb FIPA.

Els protocols d'interacció suportats per JADE son FIPA-Request, FIPA-QUery i FIPA-Propose entre d'altres. Aquests protocols s'usen per a que els agents iniciïn un protocol de conversa. Cal tenir en compte que els missatges tenen una mida definida i que si la resposta és superior a la mida establerta per el protocol, caldrà modificar la classe per a enviar la resposta en diferents paquets.

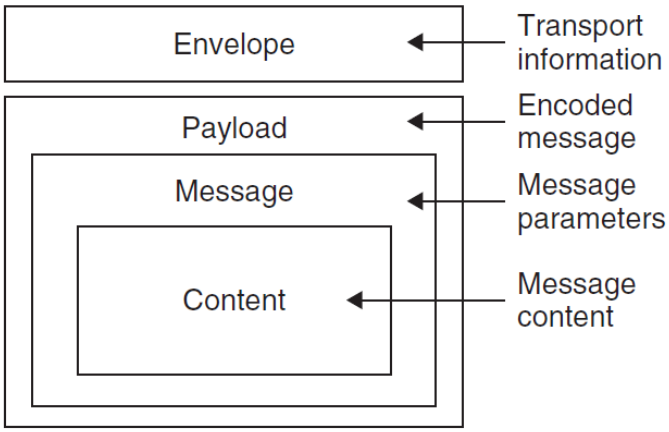


Figura 20 Estructura d'un missatge FIPA.

Alguns dels elements més importants de l'arquitectura FIPA son els següents:

- **Missatges d'agent**. Son fonamentals per a la comunicació entre agents. L'estructura d'aquests missatges és un conjunt de clau-valor escrits en FIPA-ACL. Aquests estan expressats en un llenguatge i referenciats amb una ontologia. Per a l'enviament cal que el contingut sigui codificat en un **Payload** i que sigui afegit en un **Envelope** per al protocol de transport.
- **Servei de transport de missatges**. És la manera de transportar els missatges entre agents.
- **Servei de directori d'agents**. És un repositori d'informació compartida on els agents publiquen les seves entrades de directori i on poden buscar altres agents.
- **Servei de directoris de servei**. És un repositori compartit on els agents i serveis poden descobrir altres serveis. Els serveis inclouen per exemple, serveis de transports de missatges, serveis de directoris d'agent, etc. Un servei de director de serveis també s'usa per emmagatzemar altres descripcions de serveis o serveis orientats a aplicacions.

Els missatges **FIPA-ACL** contenen un conjunt de paràmetres i aquests paràmetres poden variar segons la comunicació que es vulgui realitzar. L'únic paràmetre que és obligatori en tots els missatges ACL és el que s'anomena "**performative**". Però també s'espera que s'emplenin els camps "**sender**", "**receiver**" i altres específics (camps addicionals).

El següent fragment de codi mostra la generació d'un missatge ACL per a la comunicació entre agents:

```
ACLMessage msg = new ACLMessage(ACLMessage.REQUEST);
AID server = (AID) classifiers.next();

msg.setLanguage(codec.getName());
msg.setOntology(ontology.getName());

try {
        DataInstance dataInstance = data.get(index);
        this.dataToPredict = dataInstance;

        myAgent.getContentManager().fillContent(msg, new Action(server, dataInstance));
        msg.addReceiver(server);
        myAgent.send(msg);
        } catch (Exception ex) {
                ex.printStackTrace();
        }
}
```

Aquest segment, defineix un missatge del tipus **ACLMessage** implementant la classe *jade.lang.acl.ACLMessage* i afegeix el llenguatge, l'ontologia el receptor i les dades a enviar. Un cop el missatge s'ha composat correctament es pot enviar mitjançant la comanda **send()**.

En l'altre banda, l'agent quedarà a l'espera de rebre el missatge utilitzant un codi semblant al que es proposa a continuació:

```
ACLMessage msg = myAgent.receive(mt);
if (msg == null) {
        block();
} else {
        if (msg.getPerformative() == ACLMessage.INFORM) {
                ContentElement content = null;
                try {
                        content = getContentManager().extractContent(msg);
                } catch (Exception ex) {
                        ex.printStackTrace();
                }
                Concept action = ((Action) content).getAction();
                if (action instanceof ClassifierSettings) {
                        ClassifierSettings.put(msg.getSender().getLocalName(), action);
        }
}
```

Si a la cua de l'agent hi ha un missatge aquest es rebrà mitjançant la comanda **receive().** En el moment en que no hi hagi cap missatge, l'agent quedarà a l'espera cridant la comanda **block().**

Un altre factor a tenir en compte, és que podem filtrar els missatges a rebre, d'aquesta manera podem estar segurs que els agents només intentaran llegir missatges que vagin adreçats a ells.

Una manera molt senzilla de filtrar els tipus de missatge INFORM es pot fer de la següent manera:

```
private MessageTemplate mt = MessageTemplate.MatchPerformative(ACLMessage.INFORM);
```

Es pot també incrementar la complexitat del filtre, afegint per exemple que només s'analitzi un tipus d'ontologia o un determinat tipus de protocol.

4.4.2 Protocol de comunicació

Com hem definit anteriorment, s'usen missatges ACL per a la comunicació entre agents i la ontologia defineix que és el que volem comunicar i com volem que s'entenguin els nostres agents.

Per a la realització d'aquest projecte s'ha definit un protocol de comunicació molt simple durant la segregació de dades i que cal enviar a cada agent per al seu processament. L'agent manager enviar un tipus REQUEST a cada agent i aquests han de contestar amb una proposta del tipus PROPOSE.

Podem veure clarament com funciona aquesta comunicació mitjançant l'sniffer de Jade, ja que se'ns mostra com s'executen les comunicacions entre els agents.

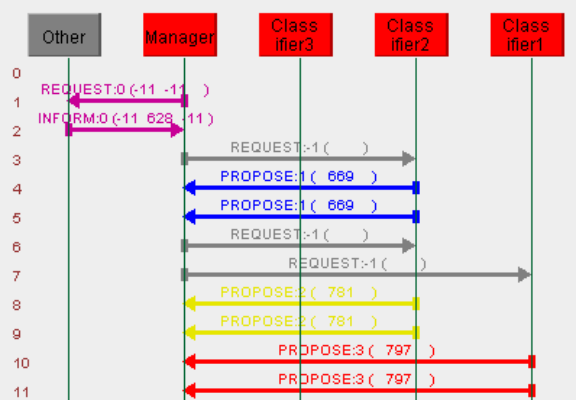


Figura 21 Comunicació entre agents.

4.4.3 Ontologies

Per a que els agents puguin comunicar entre ells, cal definir una ontologia. Aquesta, és bàsicament una representació de la informació com ho faríem mitjançant OOP. Els agents serialitzen aquesta informació i la transmeten entre agents per a que es puguin entendre entre ells.

Per a la correcta inicialització de la ontologia, cal que l'agent creï una instància d'ontologia que haurem definit anteriorment mitjançant unes eines externes com Protégé i Bean Generator que ens permetran definir quines accions i propietats volem que el nostre llenguatge tingui.

```
        private Codec codec = new SLCodec();
        private Ontology ontology = ClassifierOntology.getInstance();
        getContentManager().registerLanguage(codec);
        getContentManager().registerOntology(ontology);
```

Cal tenir present doncs, que cada vegada que vulguem enviar un missatge a un agent, caldrà definir la ontologia a usar per a que s'entenguin correctament:

```
AID server = (AID) classifiers.next();

sm.Log("Sending data to : " + server.toString());

ACLMessage msg = new ACLMessage(ACLMessage.REQUEST);
msg.setLanguage(codec.getName());
msg.setOntology(ontology.getName());
try {
        Arff_Training_Repository arff = datasetsTraining.get(i);
        myAgent.getContentManager().fillContent(msg, new Action(server, arff));
        msg.addReceiver(server);
        myAgent.send(msg);
        i++;
        System.out.println("Contacting client... Please wait!");
} catch (Exception ex) {
        ex.printStackTrace();
}
```

Exemple comunicació entre agents usant l'ontologia:

Enviament de dades de l'agent Manager a l'agent Classificador 3 amb les dades a classificar:

```
((action
     (agent-identifier
```

```
      :name Classifier3@192.168.1.85:1099/JADE
      :addresses (sequence http://jordicoll-PC:7778/acc))
    (DataInstance
      :Value "1,0.67,0,0,1,0,0.8,-1,0.25,N")))
```

Reposta de l'agent classificador a l'agent Manager amb la seva predicció:

```
((action
    (agent-identifier
      :name Manager@192.168.1.85:1099/JADE
      :addresses (sequence http://jordicoll-PC:7778/acc))
    (ClassificationResult
      :name Classifier1
      :type weka.classifiers.trees.J48
      :duration 0
      :trainingSize 64
      :numCorrect 1
      :percentage 100.0
      :instanceValue N
      :predictedInstanceValue N
      :instanceClassification 0.0
      :instancePredictedValue 0.0)))
```

Com podem veure l'enviament utilitza una acció anomenada ***DataInstance*** i la resposta una altra anomenada ***ClassificationResult***. Aquestes accions son enteses per els agents gràcies a la ontologia ***ClassifierOntology***.

Per a poder crear la ontologia d'una manera més automàtica i que després la puguem utilitzar a Jade es fa mitjançant les eines Protégé i Bean Generator.

4.4.3.1 Protégé i Bean Generator

Per la creació de la ontologia a usar, he definit aquestes mitjançant les eines Protégé i Bean Generator. Aquest paquet ens permet generar l'esquelet bàsic de la ontologia a utilitzar pels agents Jade i d'aquesta manera ens deixa especificar un tipus de llenguatge que aquests entenen per a la seva comunicació.

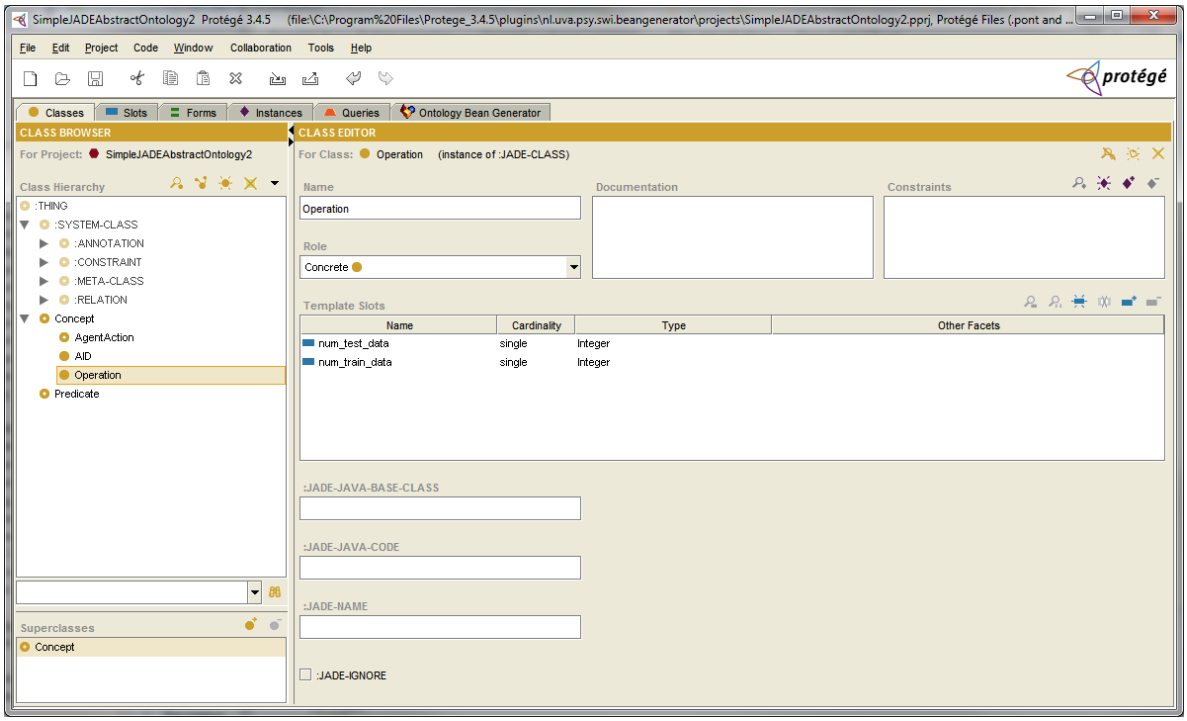


Figura 22 Exemple edició ontologia amb Protégé

Cal fer servir la versió adient del Protégé ja que les últimes versions son bastant diferents i no acaben de funcionar correctament amb el Bean Generator. Per a aquest projecte he utilitzat la versió 3.4.5 del Protégé ja que aquesta encara permet fer servir les llibreries abstractes que hi ha publicades en la pàgina del *OntologyBeanGenerator*. Un cop es segueixen les passes de la instal·lació segons es defineix a la mateixa web, llavors es pot obrir un projecte d'exemple utilitzant el SimpleJADEAbstractOntology. En aquest projecte podrem definir les propietats que necessitem i després fer servir el plugin del **BeanGenerator** per a la creació automàtica de les classes necessàries que utilitzarem en el nostre projecte.

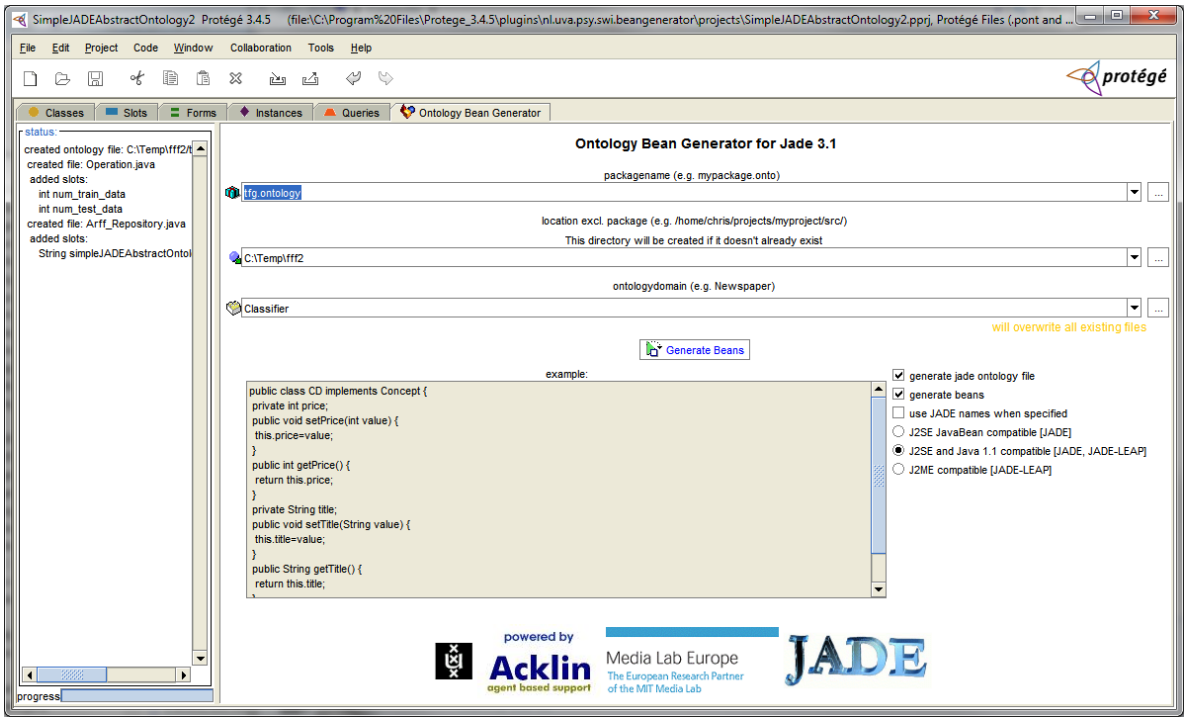


Figura 23 Exemple generació classes mitjançant Bean Generator

Un cop la generació s'ha completat, disposarem de les classes ClassifierOntology.java i Operation.java per a la seva incorporació dins el nostre projecte i que Jade serà capaç d'entendre.

Un cop tenim una estructura bàsica, podem modificar aquests fitxers i afegir les classes que creiem necessàries per a l'execució de la plataforma.

Per a aquesta ontologia s'han definit les següents accions:

- **TrainingResult**:Informa a l'agent sobre el resultat de l'entrenament que ha tingut el classificador.
- **DataInstance**: Instància de dades que el manager envia als classificador per a la seva classificació.
- **Arff_Training_Repository**: Repositori del tipus ARFF que l'agent manager envia als classificadors.
- **ClassificationResult**: Resultat de la classificació després de processar un DataInstance.
- **ManagerSettings**: Enviament de la configuració del Manager.

- **ClassifierSettings**: Enviament de la configuració del classificador.

En la següent figura es poden veure aquestes classes definides i l'ontologia:

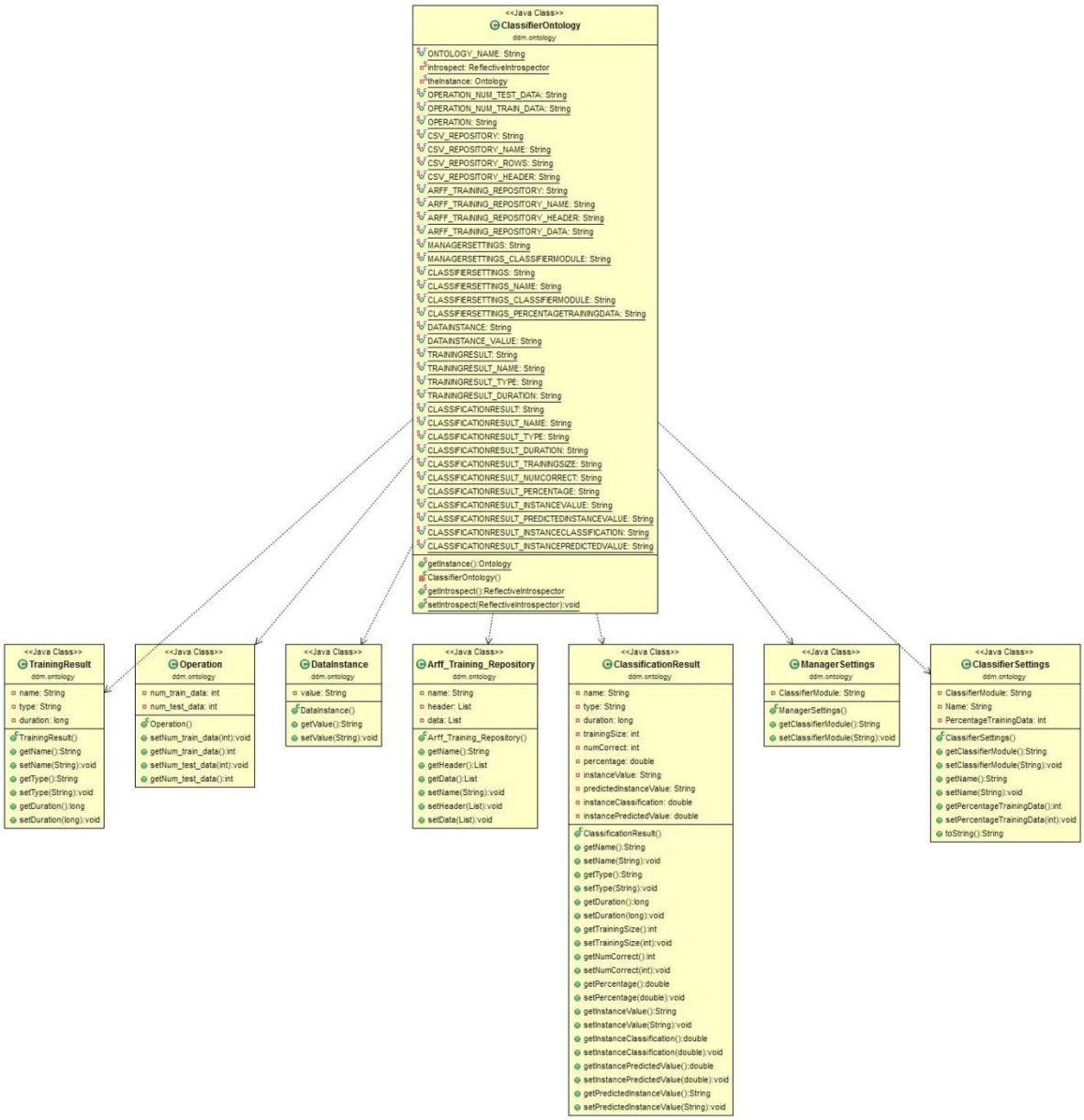


Figura 24 Diagrama classes Ontologia

Per a aquest projecte s'han definit instàncies del tipus AgentAction per la bondat del funcionament de l'aplicació ja que cada enviament requereix una acció.

4.5 Sistema de classificació

Els classificadors son una part molt important dels sistemes d'intel·ligència artificial. Aquests, utilitzen funcions o mecanismes que fan servir patrons coincidents per determinar la coincidència més propera. Els classificadors poden aprendre i classificar dades en funció de l'experiència prèvia basant-se en observacions o altres patrons.

Un classificador pot ser entrenat en diferents maneres fent servir aproximacions via aprenentatge computacional. Alguns dels més coneguts classificadors son: **Xarxes neuronals**, **El veí més proper**, **Classificació Bayes** i **arbre de decisió** entre d'altres. El rendiment de cada algorisme és diferent segons les característiques de les dades a classificar i tampoc existeix un millor algorisme. Simplement cal trobar un algorisme adient per a cada cas i anar provant diferents solucions.

Aquest projecte pot servir per comprovar aquest fet ja que s'usen 3 diferents tipus de classificadors i que es pot fer una investigació d'aprofundiment per a comprovar els resultats en funció del classificador escollit i les dades a classificar. En aquest treball es realitzarà l'estudi de dos exemples del repositori UCI i es presentaran uns resultats en funció dels algorismes utilitzats i quines han sigut les classificacions obtingudes.

Per al correcte desenvolupament d'aquest projecte i poder-nos centrar en la tasca de classificació i de presa de decisions, farem servir Weka com a eina central per a la classificació ja que disposa de tots els algorismes necessaris que volem implementar. Per tant l'organització del projecte es centra en crear uns recipients amb JADE i que aquests tinguin un mòdul weka que puguem fer servir a voluntat per a la classificació de les nostres dades.

4.5.1 La llibreria Weka

Weka és una col·lecció d'algorismes d'aprenentatge automàtic per a tasques de mineria de dades. Els algorismes bé es poden aplicar directament a un conjunt de dades o a través de crides des del propi codi Java. Weka conté eines per a les dades de pre-processament, classificació, regressió, agrupació, regles d'associació, i la visualització. També és molt adequat per al desenvolupament de nous sistemes d'aprenentatge[7].

Aquesta llibreria ens permet realitzar les nostres pròpies eines fent servir les crides de la llibreria d'una manera molt senzilla i còmode. Només cal definir uns requisits previs per a la seva utilització, tals com:

- Composar les dades correctament utilitzant capçaleres ARFF o a través d'altres formats correctament composats via fitxers CSV.
- Dades d'entrenament del classificador.
- Dades d'avaluació del classificador.
- Selecció del classificador: Generador o discriminador.
- Avaluació de les dades del classificador entrenat.

Tal com s'ha mostrat en l'apartat 3.1.1, podem realitzar la mateixa operació creant una aplicació amb Java i Weka.

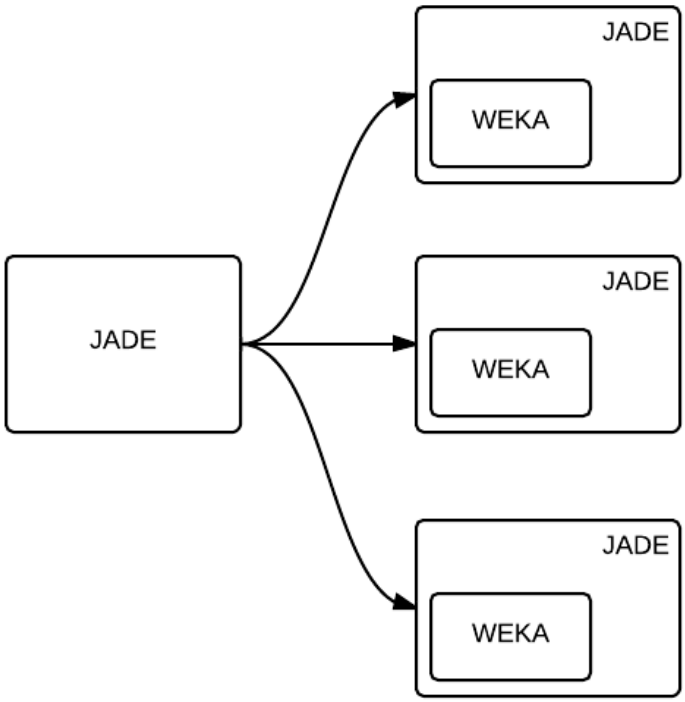


Figura 25 Embolcallament dels agents

Primer cal preparar les dades i formatar-les utilitzant capçaleres ARFF o bé composar els atributs i afegir-les en un fitxer CSV.

Per el nostre exemple, disposem de les dades actualment tractades

Training Data (trainingData.csv):

```
Season,Age,disease,Accident,Intervention,High fever,alcohol,smoking,sitting,output
-0.33,0.75,1,1,1,1,0.8,1,0.25,N
-1,0.67,1,0,0,0,1,-1,0.5,N
-0.33,0.56,1,0,0,0,1,-1,0.63,N
1,0.56,1,0,0,0,1,-1,0.44,N
0.33,0.78,1,0,0,0,1,1,0.06,N
-1,0.56,1,1,0,0,0.8,1,0.5,N
-0.33,0.92,1,0,0,1,0.6,-1,0.19,N
1,0.61,1,0,1,0,1,-1,0.63,N
-1,0.78,1,1,0,1,0.6,-1,0.38,N
-0.33,0.81,1,1,1,0,1,1,0.38,N
-1,0.61,1,0,0,0,0.8,0,0.5,N
-0.33,0.5,1,0,0,0,1,-1,0.5,N
-0.33,0.94,1,0,1,0,0.8,1,0.31,O
1,0.64,0,0,1,0,0.8,-1,0.25,N
-1,0.53,1,0,0,1,0.8,-1,0.31,N
-0.33,0.86,1,1,1,1,1,-1,0.25,N
-1,0.58,1,0,1,-1,0.8,1,0.5,N
-1,0.53,1,1,0,1,1,0,0.31,N
-1,0.78,1,0,1,0,1,-1,0.25,N
-0.33,0.83,1,1,1,0,1,-1,0.31,N
-0.33,0.67,0,0,0,-1,0.8,-1,0.44,N
-0.33,0.69,1,1,1,-1,1,-1,0.75,N
1,0.94,1,1,1,0,0.2,-1,0.25,N
-1,0.56,1,0,0,0,1,-1,0.44,N
-0.33,0.64,1,1,1,0,0.8,-1,0.31,N
-1,0.5,1,0,0,1,0.8,-1,0.44,N
-1,0.72,1,1,1,1,0.8,-1,0.19,N
-1,0.53,1,0,0,1,1,0,0.44,N
1,0.75,1,1,1,0,1,1,0.25,N
1,0.61,1,0,0,0,1,-1,0.25,N
1,0.58,0,0,0,1,0.8,1,0.44,N
-1,0.67,0,0,1,0,0.6,0,0.5,O
-1,0.81,1,1,1,1,0.8,0,0.19,N
1,0.75,1,0,0,0,0.6,0,0.25,N
-0.33,0.75,1,1,1,0,0.6,-1,0.19,N
-0.33,0.81,1,1,1,1,0.8,-1,0.38,N
1,0.75,1,1,1,0,0.8,1,0.25,N
-0.33,0.69,0,1,1,0,0.8,0,0.88,N
-0.33,0.5,1,1,0,-1,0.8,0,0.88,O
1,0.67,1,0,0,0,1,-1,0.25,N
1,0.67,1,0,0,0,0.8,1,0.38,O
-1,0.64,1,0,0,1,1,1,0.25,N
1,0.56,1,1,1,0,1,-1,0.63,N
0.33,0.75,1,0,1,0,0.8,-1,0.44,O
-0.33,0.72,1,1,0,0,0.6,1,0.19,N
-1,0.56,1,0,0,1,1,-1,0.5,N
-0.33,0.58,1,1,1,-1,0.8,0,0.19,N
1,0.56,1,0,0,1,0.6,0,0.5,N
1,0.67,1,1,0,0,0.8,-1,0.25,N
-0.33,0.53,1,1,0,0,0.8,0,0.5,N
-1,0.58,1,0,1,-1,0.8,1,0.5,N
-1,0.56,1,0,0,0,1,-1,0.44,N
-1,0.53,1,1,0,1,1,0,0.31,N
-1,0.53,1,0,0,1,1,0,0.44,N
-0.33,0.56,1,0,0,0,1,-1,0.63,N
-0.33,0.72,1,1,0,0,0.6,1,0.19,N
-0.33,0.64,1,1,1,0,0.8,-1,0.31,N
```

```
-0.33,0.75,1,1,1,0,0.6,-1,0.19,N
```

Test Data (testData.csv):

```
Season,Age,disease,Accident,Intervention,High fever,alcohol,smoking,sitting,output
-0.33,0.69,0,1,1,0,0.8,0,0.88,N
-0.33,0.94,1,0,1,0,0.8,1,0.31,O
-0.33,0.5,1,0,0,0,1,-1,0.5,N
-0.33,0.75,0,1,1,0,1,-1,0.38,N
-0.33,0.67,1,1,0,0,0.8,-1,0.5,O
-0.33,0.67,1,0,1,0,0.8,0,0.5,N
-0.33,0.67,0,0,0,-1,0.8,-1,0.44,N
-0.33,1,1,1,1,0,0.6,-1,0.38,N
1,0.64,0,0,1,0,0.8,-1,0.25,N
1,0.61,1,0,0,0,1,-1,0.25,N
1,0.67,1,1,0,-1,0.8,0,0.31,N
1,0.78,1,1,1,0,0.6,0,0.13,N
1,0.75,1,1,1,0,0.8,1,0.25,N
1,0.81,1,0,0,0,1,-1,0.38,N
```

Implementació usant Java sobre el conjunt de dades de *test* i *training*:

El següent codi font, utilitzarà un arbre de decisió (J48) i que classificarà el conjunt de dades mencionat, retornant els mateixos resultats que en la Figura 5 Execució de l'algorisme classificador J48 per agent.

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.util.Random;
import weka.classifiers.Evaluation;
import weka.classifiers.bayes.NaiveBayes;
import weka.classifiers.trees.J48;
import weka.core.Instances;
import weka.core.converters.ConverterUtils.DataSource;

public class StartWeka {
        public static void main(String[] args) throws Exception {
                //Load train data
                DataSource source = new DataSource("C:/trainingData.csv");
                Instances train = source.getDataSet();
                if (train.classIndex() == -1)
                        train.setClassIndex(train.numAttributes()-1);

                //Load test data
                DataSource source2 = new DataSource("C:/testData.csv");
                Instances test = source2.getDataSet();
                if (test.classIndex() == -1)
                        test.setClassIndex(test.numAttributes()-1);

                String[] options = new String[1];
                options[0] = "-U";              // unpruned tree
                J48 tree = new J48();           // new instance of tree
                tree.setOptions(options);       // set the options
                tree.buildClassifier(train);

                Evaluation eval = new Evaluation(train);
                eval.evaluateModel(tree, test);
                System.out.println(eval.toSummaryString("\nResults\n======\n", false));
        }
}
```

Resultat de l'execució:

```
Results
======

Correctly Classified Instances          12               85.7143 %
Incorrectly Classified Instances         2               14.2857 %
Kappa statistic                          0.4167
Mean absolute error                      0.1758
Root mean squared error                  0.3755
Relative absolute error                 82.0175 %
Root relative squared error            106.5083 %
Coverage of cases (0.95 level)          92.8571 %
Mean rel. region size (0.95 level)      75      %
Total Number of Instances               14
```

4.6 Sistema de presa de decisions

Arribats a aquest punt, cal doncs, elaborar la decisió de consens entre els agents. Per a fer-ho cal definir com es realitzarà la decisió final ja que cada classificador realitza certes classificacions amb conjunts d'entrenament de mida diferent i inclòs amb algorismes de classificació diferent. Per tant caldrà definir bé com es farà aquesta decisió basant-nos en les dades que obtenim dels agents.

L'estructura bàsica de la presa de decisió és la següent:

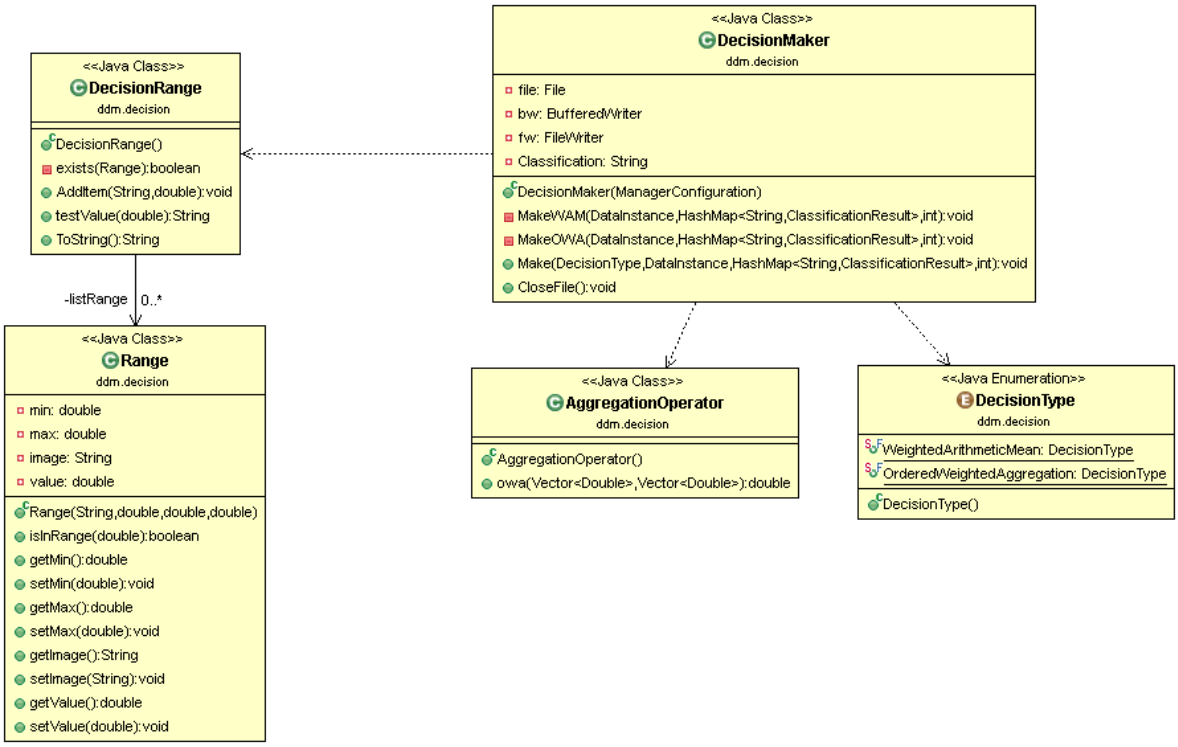


Figura 26 Disseny de les classes per a la presa de decisions

Per a la presa de decisions final m'he basat en l'utilització de l'operador OWA per al mecanisme de consens.

4.6.1 Operador OWA

L'operador OWA[8], és un operador d'agregació que proveeix una família parametritzada d'operadors d'agregació entre el mínim i el màxim. L'operador considera el grau d'importància que cada concepte té dins l'agregació i és usat en sistemes de decisió.

La definició de l'operador és la següent:

Un operador OWA de dimensió n és un mapat OWA: $R^n \rightarrow R$ que té un vector de pesos associat W de dimensió n amb $w_j \in [0,1]$ i $\sum_{j=1}^{n} w_j = 1$ tal que:

$$OWA(a_1, \ldots, a_n) = \sum_{j=1}^{n} w_j \, b_j,$$

On b_j és el *j* més gran de a_i.

També es pot fer servir el mateix operador quan els arguments estan ordenats d'acord amb el vector de pesos i aquests ordenats d'acord amb les posicions dels arguments, d'aquesta manera obtenim:

$$OWA(a_1, \ldots, a_n) = \sum_{i=1}^{n} a_i \, w_i,$$

On w_i és el *i* pes de w_j reordenat segons les posicions de a_i.

El sistema aconsegueix les respostes de tots els agents: percentatge de les dades d'entrenament com a indicador de fiabilitat i el resultat de la decisió fent servir l'algorisme de decisió que vulguin. Les dades s'agreguen i es munten en vectors per aplicar l'operador OWA.

La manera de preparar les dades és similar a la següent:

```
@Test
public void testOwa() {
        AggregationOperator ao = new AggregationOperator();
        Vector<Double> W = new Vector<Double>(3);
        Vector<Double> A = new Vector<Double>(3);
        W.addElement(0.4175);
        W.addElement(0.2216);
        W.addElement(0.3608);

        A.addElement(2.0);
        A.addElement(1.0);
        A.addElement(1.0);

        double result = ao.owa(W, A);
        System.out.println("Testing OWA " + result);
        assert(result == 1.4174);
}
```

La classe *AggregationOperator* disposa del càlcul *owa* sobre les dades en concret. En aquest mètode, es realitza l'agregació ordenada per els valors d'entrada.

4.6.2 Decisió

Arribats a aquest punt i sabent que els agents envien les dades a l'agent manager, aquests haurà de prendre una decisió amb l'ús dels mecanismes de consens disponibles. Cal dir també que el sistema està pensat per veure'n les virtuts quan es disposa d'un valor

elevat de classificadors ja que el punt de vista canvia i el consens juga un paper important.

Cal mencionar que el sistema no està pensat en quant a rendiment. Es pot composar un sistema amb 10 agents i cada agent no disposa o no poden disposar de tot el coneixement, per tant és interessant ajuntar l'opinió de diferents entitats i aconseguir un comportament força bo. Cada decisió té un temps de tractament i cal dir que s'envien valors individuals a cada agent i per tant la decisió és lenta, però molt més fiable.

El mecanisme de decisió funciona de la següent manera:

La informació de cada classificador és la següent:

Classificador 1: 81% conjunt entrenament, Decisió N
Classificador 2: 43% conjunt entrenament, Decisió O
Classificador 3: 70% conjunt entrenament, Decisió O

L'agent manager recopila aquesta informació i l'envia a la secció que combina els resultats i pren la decisió. D'aquí s'obté el vector W amb els pesos de cada classificador. Aquest vector ha de tenir un valor acumulat de 1.

Es transformen els percentatges i es crea el vector amb rang [0, 1]:

W = [0.4175, 0.2216, 0.3608]

Per obtenir aquests valors, s'ha agafat el pes de cada classificador en relació al seu conjunt d'entrenament i dividit pel total del conjunt d'entrenament dels classificadors. (81+43+70)

w_1 = 81 / (81+43+70)=0.4175

S'assignen valors numèrics a les imatges de les classificacions:

N=1 i O=0 i a partir d'aquí es crea el vector A

A = [1, 0, 0]

Un cop es tenen els 2 vectors podem passar a realitzar el càlcul de l'operador OWA:

A = [1, 0, 0]
W = [0.4175, 0.2216, 0.3608]

Es calcula l'OWA basant-nos en l'equació $\sum_{i=1}^{n} a_i\, w_i$: on els valors de w_i estan ordenats segons les entrades de a_i.

D'aquí s'obté: 1*0.4175= 0.4175.

Interpretació del resultat:

Per a ubicar aquest valor dins d'un sector, cal definir un rang. Com s'ha mencionat anteriorment, es disposa d'un valor d'imatge: N=1 i O=0. Aquests valors es defineixen directament al Weka, ja que aquest ha de donar un valor numèric automàtic a les variables de predicció de classe.

Per tant, cal fer la definició dels rangs per a ubicar el valor. En aquest cas ens guiem per uns marges. Ja que els valors son enters, podem definir un rang ±0.5 sobre el valor d'imatge. Per tant O estaria entre -0.5 i 0.5 i N entre >0.5 i 1.5.

D'aquí obtenim que el valor resultant està a 0.4174 de O i 0.5826 de N, per tant a l'estar més a prop d'O, el valor final és O.

És així doncs, que es defineix el valor final en el moment en que es passa el valor per la classe de decisió del valor final *decisionRange*:

```
AggregationOperator ao = new AggregationOperator();
double owaResult = ao.owa(W, A);
String value = decisionRange.testValue(owaResult);
if (this.Classification.length() < decisionRange.ToString().length())
        this.Classification = decisionRange.ToString();
long finishTimeMillis = System.currentTimeMillis();
long duration = (finishTimeMillis - startTimeMillis);
try {
        java.text.DecimalFormat percentageFormatter = new java.text.DecimalFormat("#0.00");
```

```
        String text = percentageFormatter.format(owaResult);
        bw.write(dataToPredict.getValue().toString() + "\t" + description + " -> Final Decision
using OWA (" + value + ") value:" + text + " in "+ duration + "ms\r\n");
} catch (Exception e) {
        e.printStackTrace();
};
```

4.6.3 Rendiment

El sistema no està pensat en quant a rendiment ja que la informació s'envia a diferents entitats i el sistema queda a l'espera de la resposta d'aquests. A més a més, la decisió del disseny de l'arquitectura fa que cada instància de dades s'enviï separadament carregant així les comunicacions ja que es podria fer un enviament massiu de dades als agents i esperar respostes conjuntes i llavors preparar l'anàlisi dins l'agent manager amb totes les respostes.
El que s'intenta aconseguir és un sistema amb un comportament força bo, amb una comunicació ràpida i evitant enviar paquets massius de dades amb el protocol de comunicacions del Jade.

Les respostes està al voltant dels 1,6s per paquets de dades petits, podent processar al voltant de 70 registres en uns 2 minuts, enviant dades a 5 agents. Durant aquest temps, el manager envia les dades als 5 agents i aquests envien la resposta, es processa i es desa a disc.

5 Proves unitàries

Dona la complexitat del projecte, han calgut una sèrie de proves unitàries com proves d'integració per tal de comprovar que el disseny de les classes és idònia i que durant els canvis constants del projecte no trenquem res i a més a més ens donen una certa seguretat i fiabilitat sobre el codi que escrivim.

El conjunt de proves unitàries és el següent:

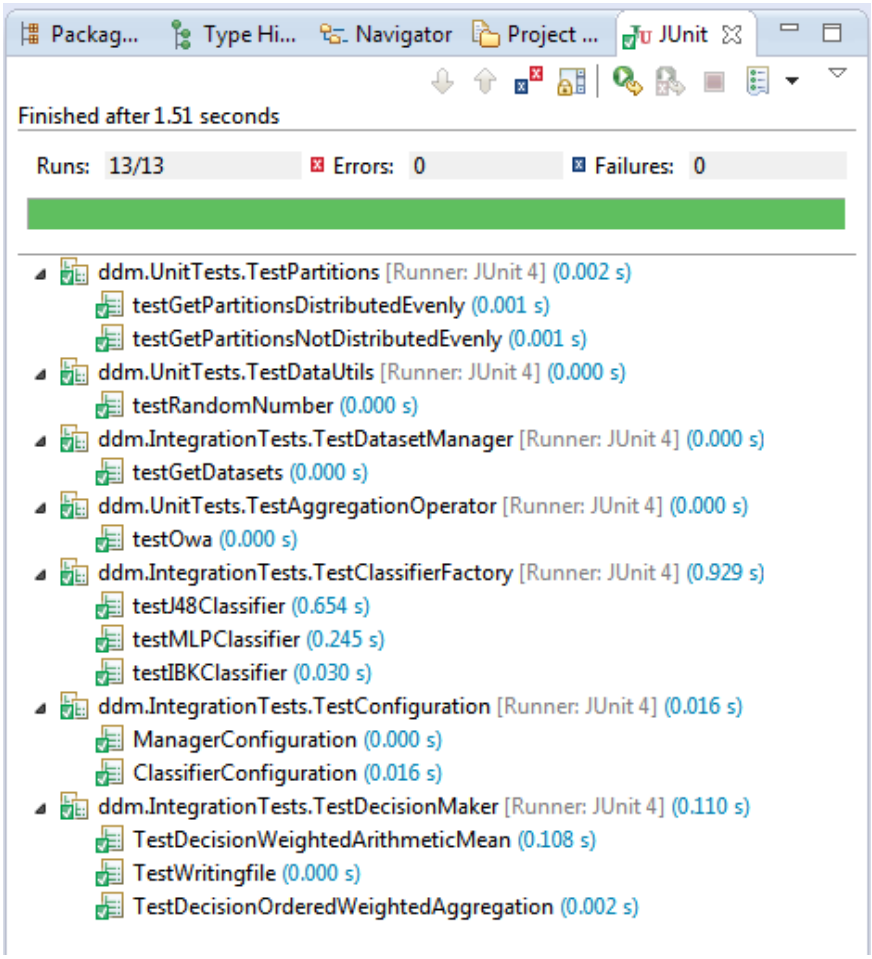


Figura 27 Conjunt de proves unitàries

Cada prova es centra en un aspecte diferent: Proves sobre els classificadors, sobre la distribució de les dades, sobre la càrrega correcta de paràmetres, sobre la decisió presa, etc. D'aquesta manera puc garantir una certa qualitat sobre el producte que lliuro.

6 Aplicació pràctica del sistema multi-agent

Per al següent estudi, s'utilitzarà el següent repositori online:

- **Iris** (https://archive.ics.uci.edu/ml/datasets/Iris).

Aquest conjunt de dades disposa de 150 tipus de flors i cal classificar-les entre *Iris Setosa*, *Iris Versicolour* i *Iris Virginica*. Utilitzarem el sistema de presa de decisions distribuït per obtenir una decisió final mitjançant combinacions de diferents algorismes classificadors i diferent mida d'entrenaments.

Primer caldrà adaptar aquest conjunt de dades a un format que el Weka pugui entendre. En aquest cas caldrà adaptar les dades a format ARFF ja que el conjunt funciona amb aquest format.

En aquesta demostració pràctica es realitzarà dos test. Un primer test utilitzant 1 agent classificador amb totes les dades d'entrenament i un segon test utilitzant 5 agents amb un número de dades d'entrenament definits o aleatoris i es compararan els resultats.

Un cop descarregat el dataset del repositori online, caldrà fer la transformació al format ARFF. Un cop feta la transformació, obtindrem un fitxer similar al següent:

```
% 1. Title: Iris Plants Database
        Updated Sept 21 by C.Blake - Added discrepency information
  %
  % 2. Sources:
  %     (a) Creator: R.A. Fisher
  %     (b) Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
  %     (c) Date: July, 1988
  %
  @RELATION iris

  @ATTRIBUTE SepalLength  NUMERIC
  @ATTRIBUTE SepalWidth  NUMERIC
  @ATTRIBUTE PetalLength  NUMERIC
  @ATTRIBUTE PetalWidth  NUMERIC
  @ATTRIBUTE output {Iris-setosa, Iris-versicolor, Iris-Virginica}

@DATA
```

5.1,3.5,1.4,0.2,Iris-setosa	7.0,3.2,4.7,1.4,Iris-versicolor	6.3,3.3,6.0,2.5,Iris-virginica
4.9,3.0,1.4,0.2,Iris-setosa	6.4,3.2,4.5,1.5,Iris-versicolor	5.8,2.7,5.1,1.9,Iris-virginica
4.7,3.2,1.3,0.2,Iris-setosa	6.9,3.1,4.9,1.5,Iris-versicolor	7.1,3.0,5.9,2.1,Iris-virginica
4.6,3.1,1.5,0.2,Iris-setosa	5.5,2.3,4.0,1.3,Iris-versicolor	6.3,2.9,5.6,1.8,Iris-virginica
5.0,3.6,1.4,0.2,Iris-setosa	6.5,2.8,4.6,1.5,Iris-versicolor	6.5,3.0,5.8,2.2,Iris-virginica
5.4,3.9,1.7,0.4,Iris-setosa	5.7,2.8,4.5,1.3,Iris-versicolor	7.6,3.0,6.6,2.1,Iris-virginica
4.6,3.4,1.4,0.3,Iris-setosa	6.3,3.3,4.7,1.6,Iris-versicolor	4.9,2.5,4.5,1.7,Iris-virginica
5.0,3.4,1.5,0.2,Iris-setosa	4.9,2.4,3.3,1.0,Iris-versicolor	7.3,2.9,6.3,1.8,Iris-virginica
4.4,2.9,1.4,0.2,Iris-setosa	6.6,2.9,4.6,1.3,Iris-versicolor	6.7,2.5,5.8,1.8,Iris-virginica
4.9,3.1,1.5,0.1,Iris-setosa	5.2,2.7,3.9,1.4,Iris-versicolor	7.2,3.6,6.1,2.5,Iris-virginica
5.4,3.7,1.5,0.2,Iris-setosa	5.0,2.0,3.5,1.0,Iris-versicolor	6.5,3.2,5.1,2.0,Iris-virginica
4.8,3.4,1.6,0.2,Iris-setosa	5.9,3.0,4.2,1.5,Iris-versicolor	6.4,2.7,5.3,1.9,Iris-virginica
4.8,3.0,1.4,0.1,Iris-setosa	6.0,2.2,4.0,1.0,Iris-versicolor	6.8,3.0,5.5,2.1,Iris-virginica
4.3,3.0,1.1,0.1,Iris-setosa	6.1,2.9,4.7,1.4,Iris-versicolor	5.7,2.5,5.0,2.0,Iris-virginica
5.8,4.0,1.2,0.2,Iris-setosa	5.6,2.9,3.6,1.3,Iris-versicolor	5.8,2.8,5.1,2.4,Iris-virginica
5.7,4.4,1.5,0.4,Iris-setosa	6.7,3.1,4.4,1.4,Iris-versicolor	6.4,3.2,5.3,2.3,Iris-virginica
5.4,3.9,1.3,0.4,Iris-setosa	5.6,3.0,4.5,1.5,Iris-versicolor	6.5,3.0,5.5,1.8,Iris-virginica
5.1,3.5,1.4,0.3,Iris-setosa	5.8,2.7,4.1,1.0,Iris-versicolor	7.7,3.8,6.7,2.2,Iris-virginica
5.7,3.8,1.7,0.3,Iris-setosa	6.2,2.2,4.5,1.5,Iris-versicolor	7.7,2.6,6.9,2.3,Iris-virginica
5.1,3.8,1.5,0.3,Iris-setosa	5.6,2.5,3.9,1.1,Iris-versicolor	6.0,2.2,5.0,1.5,Iris-virginica
5.4,3.4,1.7,0.2,Iris-setosa	5.9,3.2,4.8,1.8,Iris-versicolor	6.9,3.2,5.7,2.3,Iris-virginica
5.1,3.7,1.5,0.4,Iris-setosa	6.1,2.8,4.0,1.3,Iris-versicolor	5.6,2.8,4.9,2.0,Iris-virginica
4.6,3.6,1.0,0.2,Iris-setosa	6.3,2.5,4.9,1.5,Iris-versicolor	7.7,2.8,6.7,2.0,Iris-virginica
5.1,3.3,1.7,0.5,Iris-setosa	6.1,2.8,4.7,1.2,Iris-versicolor	6.3,2.7,4.9,1.8,Iris-virginica
4.8,3.4,1.9,0.2,Iris-setosa	6.4,2.9,4.3,1.3,Iris-versicolor	6.7,3.3,5.7,2.1,Iris-virginica
5.0,3.0,1.6,0.2,Iris-setosa	6.6,3.0,4.4,1.4,Iris-versicolor	7.2,3.2,6.0,1.8,Iris-virginica
5.0,3.4,1.6,0.4,Iris-setosa	6.8,2.8,4.8,1.4,Iris-versicolor	6.2,2.8,4.8,1.8,Iris-virginica
5.2,3.5,1.5,0.2,Iris-setosa	6.7,3.0,5.0,1.7,Iris-versicolor	6.1,3.0,4.9,1.8,Iris-virginica
5.2,3.4,1.4,0.2,Iris-setosa	6.0,2.9,4.5,1.5,Iris-versicolor	6.4,2.8,5.6,2.1,Iris-virginica
4.7,3.2,1.6,0.2,Iris-setosa	5.7,2.6,3.5,1.0,Iris-versicolor	7.2,3.0,5.8,1.6,Iris-virginica
4.8,3.1,1.6,0.2,Iris-setosa	5.5,2.4,3.8,1.1,Iris-versicolor	7.4,2.8,6.1,1.9,Iris-virginica
5.4,3.4,1.5,0.4,Iris-setosa	5.5,2.4,3.7,1.0,Iris-versicolor	7.9,3.8,6.4,2.0,Iris-virginica
5.2,4.1,1.5,0.1,Iris-setosa	5.8,2.7,3.9,1.2,Iris-versicolor	6.4,2.8,5.6,2.2,Iris-virginica
5.5,4.2,1.4,0.2,Iris-setosa	6.0,2.7,5.1,1.6,Iris-versicolor	6.3,2.8,5.1,1.5,Iris-virginica
4.9,3.1,1.5,0.1,Iris-setosa	5.4,3.0,4.5,1.5,Iris-versicolor	6.1,2.6,5.6,1.4,Iris-virginica
5.0,3.2,1.2,0.2,Iris-setosa	6.0,3.4,4.5,1.6,Iris-versicolor	7.7,3.0,6.1,2.3,Iris-virginica
5.5,3.5,1.3,0.2,Iris-setosa	6.7,3.1,4.7,1.5,Iris-versicolor	6.3,3.4,5.6,2.4,Iris-virginica
4.9,3.1,1.5,0.1,Iris-setosa	6.3,2.3,4.4,1.3,Iris-versicolor	6.4,3.1,5.5,1.8,Iris-virginica
4.4,3.0,1.3,0.2,Iris-setosa	5.6,3.0,4.1,1.3,Iris-versicolor	6.0,3.0,4.8,1.8,Iris-virginica
5.1,3.4,1.5,0.2,Iris-setosa	5.5,2.5,4.0,1.3,Iris-versicolor	6.9,3.1,5.4,2.1,Iris-virginica
5.0,3.5,1.3,0.3,Iris-setosa	5.5,2.6,4.4,1.2,Iris-versicolor	6.7,3.1,5.6,2.4,Iris-virginica
4.5,2.3,1.3,0.3,Iris-setosa	6.1,3.0,4.6,1.4,Iris-versicolor	6.9,3.1,5.1,2.3,Iris-virginica
4.4,3.2,1.3,0.2,Iris-setosa	5.8,2.6,4.0,1.2,Iris-versicolor	5.8,2.7,5.1,1.9,Iris-virginica
5.0,3.5,1.6,0.6,Iris-setosa	5.0,2.3,3.3,1.0,Iris-versicolor	6.8,3.2,5.9,2.3,Iris-virginica
5.1,3.8,1.9,0.4,Iris-setosa	5.6,2.7,4.2,1.3,Iris-versicolor	6.7,3.3,5.7,2.5,Iris-virginica

4.8,3.0,1.4,0.3,Iris-setosa	5.7,3.0,4.2,1.2,Iris-versicolor	6.7,3.0,5.2,2.3,Iris-virginica
5.1,3.8,1.6,0.2,Iris-setosa	5.7,2.9,4.2,1.3,Iris-versicolor	6.3,2.5,5.0,1.9,Iris-virginica
4.6,3.2,1.4,0.2,Iris-setosa	6.2,2.9,4.3,1.3,Iris-versicolor	6.5,3.0,5.2,2.0,Iris-virginica
5.3,3.7,1.5,0.2,Iris-setosa	5.1,2.5,3.0,1.1,Iris-versicolor	6.2,3.4,5.4,2.3,Iris-virginica
5.0,3.3,1.4,0.2,Iris-setosa	5.7,2.8,4.1,1.3,Iris-versicolor	5.9,3.0,5.1,1.8,Iris-virginica

Aquest conjunt el podem fer passar per el weka, per comprovar que l'estructura és correcte i fer una classificació inicial per veure resultats:

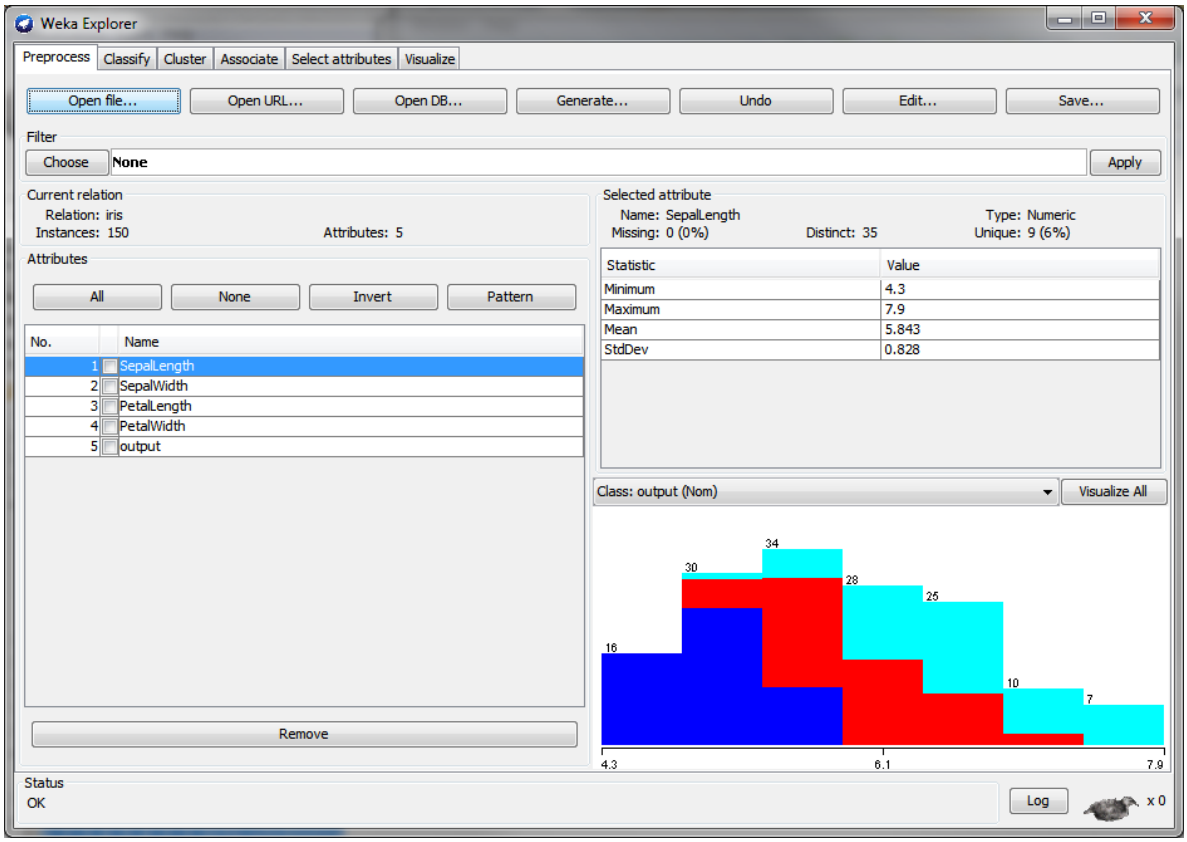


Figura 28 Anàlisi amb weka

D'aquí podem obtenir una classificació inicial mitjançant qualsevol dels classificadors existents:

```
=== Run information ===

Scheme:weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     iris
Instances:    150
Attributes:   5
              SepalLength
```

```
            SepalWidth
            PetalLength
            PetalWidth
            output
Test mode:10-fold cross-validation

=== Classifier model (full training set) ===

J48 pruned tree
------------------

PetalWidth <= 0.6: Iris-setosa (50.0)
PetalWidth > 0.6
|   PetalWidth <= 1.7
|   |   PetalLength <= 4.9: Iris-versicolor (48.0/1.0)
|   |   PetalLength > 4.9
|   |   |   PetalWidth <= 1.5: Iris-virginica (3.0)
|   |   |   PetalWidth > 1.5: Iris-versicolor (3.0/1.0)
|   PetalWidth > 1.7: Iris-virginica (46.0/1.0)

Number of Leaves  :             5

Size of the tree :      9


Time taken to build model: 0.04 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         144               96      %
Incorrectly Classified Instances         6               4       %
Kappa statistic                          0.94
Mean absolute error                      0.035
Root mean squared error                  0.1586
Relative absolute error                  7.8705 %
Root relative squared error             33.6353 %
Total Number of Instances              150

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.98      0          1        0.98      0.99       0.99     Iris-setosa
                 0.94      0.03       0.94      0.94      0.94       0.952    Iris-versicolor
                 0.96      0.03       0.941     0.96      0.95       0.961    Iris-virginica
Weighted Avg.    0.96      0.02       0.96      0.96      0.96       0.968
```

```
=== Confusion Matrix ===

 a  b  c   <-- classified as
49  1  0 |  a = Iris-setosa
 0 47  3 |  b = Iris-versicolor
 0  2 48 |  c = Iris-virginica
```

6.1 Test 1. 1 Agent Classificador

Per a l'execució d'aquest test, primer s'editarà el fitxer d'entrada per organitzar millor les dades ja que aquest fitxer és el que s'utilitzarà per definir el conjunt d'entrenament. En la definició inicial, el fitxer conté 3 grups amb les diferents classificacions, ara bé caldrà organitzar el fitxer per a que la classificació estigui desordenada i així poder muntar un conjunt d'entrenament molt més fiable:

5.1,3.5,1.4,0.2,Iris-setosa	6.5,3.0,5.5,1.8,Iris-virginica	6.0,2.7,5.1,1.6,Iris-versicolor
7.0,3.2,4.7,1.4,Iris-versicolor	5.1,3.5,1.4,0.3,Iris-setosa	6.3,2.8,5.1,1.5,Iris-virginica
6.3,3.3,6.0,2.5,Iris-virginica	5.8,2.7,4.1,1.0,Iris-versicolor	4.9,3.1,1.5,0.1,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa	7.7,3.8,6.7,2.2,Iris-virginica	5.4,3.0,4.5,1.5,Iris-versicolor
6.4,3.2,4.5,1.5,Iris-versicolor	5.7,3.8,1.7,0.3,Iris-setosa	6.1,2.6,5.6,1.4,Iris-virginica
5.8,2.7,5.1,1.9,Iris-virginica	6.2,2.2,4.5,1.5,Iris-versicolor	5.0,3.2,1.2,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa	7.7,2.6,6.9,2.3,Iris-virginica	6.0,3.4,4.5,1.6,Iris-versicolor
6.9,3.1,4.9,1.5,Iris-versicolor	5.1,3.8,1.5,0.3,Iris-setosa	7.7,3.0,6.1,2.3,Iris-virginica
7.1,3.0,5.9,2.1,Iris-virginica	5.6,2.5,3.9,1.1,Iris-versicolor	5.5,3.5,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa	6.0,2.2,5.0,1.5,Iris-virginica	6.7,3.1,4.7,1.5,Iris-versicolor
5.5,2.3,4.0,1.3,Iris-versicolor	5.4,3.4,1.7,0.2,Iris-setosa	6.3,3.4,5.6,2.4,Iris-virginica
6.3,2.9,5.6,1.8,Iris-virginica	5.9,3.2,4.8,1.8,Iris-versicolor	4.9,3.1,1.5,0.1,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa	6.9,3.2,5.7,2.3,Iris-virginica	6.3,2.3,4.4,1.3,Iris-versicolor
6.5,2.8,4.6,1.5,Iris-versicolor	5.1,3.7,1.5,0.4,Iris-setosa	6.4,3.1,5.5,1.8,Iris-virginica
6.5,3.0,5.8,2.2,Iris-virginica	6.1,2.8,4.0,1.3,Iris-versicolor	4.4,3.0,1.3,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa	5.6,2.8,4.9,2.0,Iris-virginica	5.6,3.0,4.1,1.3,Iris-versicolor
5.7,2.8,4.5,1.3,Iris-versicolor	4.6,3.6,1.0,0.2,Iris-setosa	6.0,3.0,4.8,1.8,Iris-virginica
7.6,3.0,6.6,2.1,Iris-virginica	6.3,2.5,4.9,1.5,Iris-versicolor	5.1,3.4,1.5,0.2,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa	7.7,2.8,6.7,2.0,Iris-virginica	5.5,2.5,4.0,1.3,Iris-versicolor
6.3,3.3,4.7,1.6,Iris-versicolor	5.1,3.3,1.7,0.5,Iris-setosa	6.9,3.1,5.4,2.1,Iris-virginica
4.9,2.5,4.5,1.7,Iris-virginica	6.1,2.8,4.7,1.2,Iris-versicolor	5.0,3.5,1.3,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa	6.3,2.7,4.9,1.8,Iris-virginica	5.5,2.6,4.4,1.2,Iris-versicolor
4.9,2.4,3.3,1.0,Iris-versicolor	4.8,3.4,1.9,0.2,Iris-setosa	6.7,3.1,5.6,2.4,Iris-virginica
7.3,2.9,6.3,1.8,Iris-virginica	6.4,2.9,4.3,1.3,Iris-versicolor	4.5,2.3,1.3,0.3,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa	6.7,3.3,5.7,2.1,Iris-virginica	6.1,3.0,4.6,1.4,Iris-versicolor
6.6,2.9,4.6,1.3,Iris-versicolor	5.0,3.0,1.6,0.2,Iris-setosa	6.9,3.1,5.1,2.3,Iris-virginica
6.7,2.5,5.8,1.8,Iris-virginica	6.6,3.0,4.4,1.4,Iris-versicolor	4.4,3.2,1.3,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa	7.2,3.2,6.0,1.8,Iris-virginica	5.8,2.6,4.0,1.2,Iris-versicolor
5.2,2.7,3.9,1.4,Iris-versicolor	5.0,3.4,1.6,0.4,Iris-setosa	5.8,2.7,5.1,1.9,Iris-virginica

7.2,3.6,6.1,2.5,Iris-virginica	6.8,2.8,4.8,1.4,Iris-versicolor	5.0,3.5,1.6,0.6,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa	6.2,2.8,4.8,1.8,Iris-virginica	5.0,2.3,3.3,1.0,Iris-versicolor
5.0,2.0,3.5,1.0,Iris-versicolor	5.2,3.5,1.5,0.2,Iris-setosa	6.8,3.2,5.9,2.3,Iris-virginica
6.5,3.2,5.1,2.0,Iris-virginica	6.7,3.0,5.0,1.7,Iris-versicolor	5.1,3.8,1.9,0.4,Iris-setosa
4.8,3.4,1.6,0.2,Iris-setosa	6.1,3.0,4.9,1.8,Iris-virginica	5.6,2.7,4.2,1.3,Iris-versicolor
5.9,3.0,4.2,1.5,Iris-versicolor	5.2,3.4,1.4,0.2,Iris-setosa	6.7,3.3,5.7,2.5,Iris-virginica
6.4,2.7,5.3,1.9,Iris-virginica	6.0,2.9,4.5,1.5,Iris-versicolor	4.8,3.0,1.4,0.3,Iris-setosa
4.8,3.0,1.4,0.1,Iris-setosa	6.4,2.8,5.6,2.1,Iris-virginica	5.7,3.0,4.2,1.2,Iris-versicolor
6.0,2.2,4.0,1.0,Iris-versicolor	4.7,3.2,1.6,0.2,Iris-setosa	6.7,3.0,5.2,2.3,Iris-virginica
6.8,3.0,5.5,2.1,Iris-virginica	5.7,2.6,3.5,1.0,Iris-versicolor	5.1,3.8,1.6,0.2,Iris-setosa
4.3,3.0,1.1,0.1,Iris-setosa	7.2,3.0,5.8,1.6,Iris-virginica	5.7,2.9,4.2,1.3,Iris-versicolor
6.1,2.9,4.7,1.4,Iris-versicolor	4.8,3.1,1.6,0.2,Iris-setosa	6.3,2.5,5.0,1.9,Iris-virginica
5.7,2.5,5.0,2.0,Iris-virginica	5.5,2.4,3.8,1.1,Iris-versicolor	4.6,3.2,1.4,0.2,Iris-setosa
5.8,4.0,1.2,0.2,Iris-setosa	7.4,2.8,6.1,1.9,Iris-virginica	6.2,2.9,4.3,1.3,Iris-versicolor
5.6,2.9,3.6,1.3,Iris-versicolor	5.4,3.4,1.5,0.4,Iris-setosa	6.5,3.0,5.2,2.0,Iris-virginica
5.8,2.8,5.1,2.4,Iris-virginica	5.5,2.4,3.7,1.0,Iris-versicolor	5.3,3.7,1.5,0.2,Iris-setosa
5.7,4.4,1.5,0.4,Iris-setosa	7.9,3.8,6.4,2.0,Iris-virginica	5.1,2.5,3.0,1.1,Iris-versicolor
6.7,3.1,4.4,1.4,Iris-versicolor	5.2,4.1,1.5,0.1,Iris-setosa	6.2,3.4,5.4,2.3,Iris-virginica
6.4,3.2,5.3,2.3,Iris-virginica	5.8,2.7,3.9,1.2,Iris-versicolor	5.0,3.3,1.4,0.2,Iris-setosa
5.4,3.9,1.3,0.4,Iris-setosa	6.4,2.8,5.6,2.2,Iris-virginica	5.7,2.8,4.1,1.3,Iris-versicolor
5.6,3.0,4.5,1.5,Iris-versicolor	5.5,4.2,1.4,0.2,Iris-setosa	5.9,3.0,5.1,1.8,Iris-virginica

D'aquest conjunt de dades es triarà el 50% com a conjunt d'entrenament i la resta es passarà per el classificador. Es configurarà el sistema de la següent manera i amb els següents valors:

Manager:

```
#Manager Properties
#Location of the data
ArffDataSetLocation=C:/workspace/TFG-DistributedDecisionJade/data/iris.arff
#Percentage of the data dedicated to training
PercentageTrainingData=50
NumberOfClassifiers=1
#Classifier to use J48, IBk, MLP. Put "random" so the classifier can choose one randomly
ClassifierModule1=IBk
#Classifier uses the same Jade container true/false
ClassifiersUseSameContainer=false
#Output file after the decision has been made
OutputFileLocation=C:/workspace/TFG-DistributedDecisionJade/src/ddm/Output/iris_Diagnosis.out
#Verbosity true/false if we want the tool to add extra information on the screen
Verbosity=true
#Logging true/false to enable logging using log4j
Logging=true
```

Classifier:

```
#Classifier Properties
ApplicationPath=C:/workspace/TFG-DistributedDecisionJade/bin/
#Minimum and Maximum percentage of the data to use as a training
MinimumPercentageTraining=100
MaximumPercentageTraining=100
#Verbosity true/false if we want the tool to add extra information on the screen
Verbosity=true
#Logging true/false to enable logging using log4j
Logging=true
```

Definim que volem el 100% de les dades destinades a l'entrenament.

Ara executem l'aplicació i marquem les opcions al terminal. Primer creem els agents, després enviem les dades d'entrenament i llavors executem les dades a classificar. La resposta final es trobarà en el directori de sortida que hem definit en les propietats de l'agent.

D'aquesta execució finalment obtenim:

```
*******************************************************************************
This file has been auto generated by DDM Distributed Decision Making System
*******************************************************************************

5.0,3.0,1.6,0.2,Iris-setosa  Classifier1 (weka.classifiers.lazy.IBk) 4ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 4ms
6.6,3.0,4.4,1.4,Iris-versicolor              Classifier1 (weka.classifiers.lazy.IBk) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 1ms
7.2,3.2,6.0,1.8,Iris-virginica               Classifier1 (weka.classifiers.lazy.IBk) 1ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
5.0,3.4,1.6,0.4,Iris-setosa  Classifier1 (weka.classifiers.lazy.IBk) 0ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
6.8,2.8,4.8,1.4,Iris-versicolor              Classifier1 (weka.classifiers.lazy.IBk) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
6.2,2.8,4.8,1.8,Iris-virginica               Classifier1 (weka.classifiers.lazy.IBk) 0ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
5.2,3.5,1.5,0.2,Iris-setosa  Classifier1 (weka.classifiers.lazy.IBk) 1ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
6.7,3.0,5.0,1.7,Iris-versicolor              Classifier1 (weka.classifiers.lazy.IBk) 1ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms **disagreed
6.1,3.0,4.9,1.8,Iris-virginica               Classifier1 (weka.classifiers.lazy.IBk) 1ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 1ms **disagreed
5.2,3.4,1.4,0.2,Iris-setosa  Classifier1 (weka.classifiers.lazy.IBk) 0ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
6.0,2.9,4.5,1.5,Iris-versicolor              Classifier1 (weka.classifiers.lazy.IBk) 1ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
6.4,2.8,5.6,2.1,Iris-virginica               Classifier1 (weka.classifiers.lazy.IBk) 1ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
4.7,3.2,1.6,0.2,Iris-setosa  Classifier1 (weka.classifiers.lazy.IBk) 0ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
5.7,2.6,3.5,1.0,Iris-versicolor              Classifier1 (weka.classifiers.lazy.IBk) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
7.2,3.0,5.8,1.6,Iris-virginica               Classifier1 (weka.classifiers.lazy.IBk) 0ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
4.8,3.1,1.6,0.2,Iris-setosa  Classifier1 (weka.classifiers.lazy.IBk) 1ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
5.5,2.4,3.8,1.1,Iris-versicolor              Classifier1 (weka.classifiers.lazy.IBk) 1ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
7.4,2.8,6.1,1.9,Iris-virginica               Classifier1 (weka.classifiers.lazy.IBk) 1ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
5.4,3.4,1.5,0.4,Iris-setosa  Classifier1 (weka.classifiers.lazy.IBk) 1ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
5.5,2.4,3.7,1.0,Iris-versicolor              Classifier1 (weka.classifiers.lazy.IBk) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
7.9,3.8,6.4,2.0,Iris-virginica               Classifier1 (weka.classifiers.lazy.IBk) 1ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
5.2,4.1,1.5,0.1,Iris-setosa  Classifier1 (weka.classifiers.lazy.IBk) 1ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
5.8,2.7,3.9,1.2,Iris-versicolor              Classifier1 (weka.classifiers.lazy.IBk) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
6.4,2.8,5.6,2.2,Iris-virginica               Classifier1 (weka.classifiers.lazy.IBk) 0ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
5.5,4.2,1.4,0.2,Iris-setosa  Classifier1 (weka.classifiers.lazy.IBk) 0ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 1ms
6.0,2.7,5.1,1.6,Iris-versicolor              Classifier1 (weka.classifiers.lazy.IBk) 0ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms **disagreed
6.3,2.8,5.1,1.5,Iris-virginica               Classifier1 (weka.classifiers.lazy.IBk) 1ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms **disagreed
4.9,3.1,1.5,0.1,Iris-setosa  Classifier1 (weka.classifiers.lazy.IBk) 1ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
5.4,3.0,4.5,1.5,Iris-versicolor              Classifier1 (weka.classifiers.lazy.IBk) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
6.1,2.6,5.6,1.4,Iris-virginica               Classifier1 (weka.classifiers.lazy.IBk) 1ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 1ms **disagreed
5.0,3.2,1.2,0.2,Iris-setosa  Classifier1 (weka.classifiers.lazy.IBk) 1ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
6.0,3.4,4.5,1.6,Iris-versicolor              Classifier1 (weka.classifiers.lazy.IBk) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
7.7,3.0,6.1,2.3,Iris-virginica               Classifier1 (weka.classifiers.lazy.IBk) 1ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
5.5,3.5,1.3,0.2,Iris-setosa  Classifier1 (weka.classifiers.lazy.IBk) 0ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
6.7,3.1,4.7,1.5,Iris-versicolor              Classifier1 (weka.classifiers.lazy.IBk) 1ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
6.3,3.4,5.6,2.4,Iris-virginica               Classifier1 (weka.classifiers.lazy.IBk) 0ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
4.9,3.1,1.5,0.1,Iris-setosa  Classifier1 (weka.classifiers.lazy.IBk) 0ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
6.3,2.3,4.4,1.3,Iris-versicolor              Classifier1 (weka.classifiers.lazy.IBk) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
6.4,3.1,5.5,1.8,Iris-virginica               Classifier1 (weka.classifiers.lazy.IBk) 1ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 1ms
```

```
4.4,3.0,1.3,0.2,Iris-setosa  Classifier1 (weka.classifiers.lazy.IBk) 1ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
5.6,3.0,4.1,1.3,Iris-versicolor  Classifier1 (weka.classifiers.lazy.IBk) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
6.0,3.0,4.8,1.8,Iris-virginica  Classifier1 (weka.classifiers.lazy.IBk) 1ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms **disagreed
5.1,3.4,1.5,0.2,Iris-setosa  Classifier1 (weka.classifiers.lazy.IBk) 1ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
5.5,2.5,4.0,1.3,Iris-versicolor  Classifier1 (weka.classifiers.lazy.IBk) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
6.9,3.1,5.4,2.1,Iris-virginica  Classifier1 (weka.classifiers.lazy.IBk) 3ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
5.0,3.5,1.3,0.3,Iris-setosa  Classifier1 (weka.classifiers.lazy.IBk) 0ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
5.5,2.6,4.4,1.2,Iris-versicolor  Classifier1 (weka.classifiers.lazy.IBk) 1ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 1ms
6.7,3.1,5.6,2.4,Iris-virginica  Classifier1 (weka.classifiers.lazy.IBk) 0ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
4.5,2.3,1.3,0.3,Iris-setosa  Classifier1 (weka.classifiers.lazy.IBk) 0ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
6.1,3.0,4.6,1.4,Iris-versicolor  Classifier1 (weka.classifiers.lazy.IBk) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
6.9,3.1,5.1,2.3,Iris-virginica  Classifier1 (weka.classifiers.lazy.IBk) 1ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
4.4,3.2,1.3,0.2,Iris-setosa  Classifier1 (weka.classifiers.lazy.IBk) 1ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
5.8,2.6,4.0,1.2,Iris-versicolor  Classifier1 (weka.classifiers.lazy.IBk) 1ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
5.8,2.7,5.1,1.9,Iris-virginica  Classifier1 (weka.classifiers.lazy.IBk) 0ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
5.0,3.5,1.6,0.6,Iris-setosa  Classifier1 (weka.classifiers.lazy.IBk) 0ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
5.0,2.3,3.3,1.0,Iris-versicolor  Classifier1 (weka.classifiers.lazy.IBk) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
6.8,3.2,5.9,2.3,Iris-virginica  Classifier1 (weka.classifiers.lazy.IBk) 0ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
5.1,3.8,1.9,0.4,Iris-setosa  Classifier1 (weka.classifiers.lazy.IBk) 0ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
5.6,2.7,4.2,1.3,Iris-versicolor  Classifier1 (weka.classifiers.lazy.IBk) 1ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 1ms
6.7,3.3,5.7,2.5,Iris-virginica  Classifier1 (weka.classifiers.lazy.IBk) 1ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 1ms
4.8,3.0,1.4,0.3,Iris-setosa  Classifier1 (weka.classifiers.lazy.IBk) 1ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
5.7,3.0,4.2,1.2,Iris-versicolor  Classifier1 (weka.classifiers.lazy.IBk) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
6.7,3.0,5.2,2.3,Iris-virginica  Classifier1 (weka.classifiers.lazy.IBk) 1ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
5.1,3.8,1.6,0.2,Iris-setosa  Classifier1 (weka.classifiers.lazy.IBk) 0ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
5.7,2.9,4.2,1.3,Iris-versicolor  Classifier1 (weka.classifiers.lazy.IBk) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
6.3,2.5,5.0,1.9,Iris-virginica  Classifier1 (weka.classifiers.lazy.IBk) 0ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
4.6,3.2,1.4,0.2,Iris-setosa  Classifier1 (weka.classifiers.lazy.IBk) 0ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
6.2,2.9,4.3,1.3,Iris-versicolor  Classifier1 (weka.classifiers.lazy.IBk) 1ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
6.5,3.0,5.2,2.0,Iris-virginica  Classifier1 (weka.classifiers.lazy.IBk) 1ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
5.3,3.7,1.5,0.2,Iris-setosa  Classifier1 (weka.classifiers.lazy.IBk) 0ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
5.1,2.5,3.0,1.1,Iris-versicolor  Classifier1 (weka.classifiers.lazy.IBk) 1ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
6.2,3.4,5.4,2.3,Iris-virginica  Classifier1 (weka.classifiers.lazy.IBk) 1ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
5.0,3.3,1.4,0.2,Iris-setosa  Classifier1 (weka.classifiers.lazy.IBk) 0ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
5.7,2.8,4.1,1.3,Iris-versicolor  Classifier1 (weka.classifiers.lazy.IBk) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms

**************************************************************************
Image: Iris-setosa associated value: 0.0
Image: Iris-versicolor associated value: 1.0
Image: Iris-virginica associated value: 2.0
**************************************************************************
Duration 84785ms
```

On s'indica que hi ha 6 elements que discrepen amb els resultats de la classificació inicial del repositori i el sistema mostra una nova classificació després de fer-lo passar per el sistema.

El sistema triga 84s a processar aquests 50 registres i a prendre una decisió.

Per poder veure una diferència clara, podem executar el mateix test, fent servir un altre classificador i comparar els resultats:

```
**************************************************************************
This file has been auto generated by DDM Distributed Decision Making System
**************************************************************************

5.0,3.0,1.6,0.2,Iris-setosa  Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 111ms
6.6,3.0,4.4,1.4,Iris-versicolor  Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 1ms
7.2,3.2,6.0,1.8,Iris-virginica  Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
5.0,3.4,1.6,0.4,Iris-setosa  Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
6.8,2.8,4.8,1.4,Iris-versicolor  Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 1ms
6.2,2.8,4.8,1.8,Iris-virginica  Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
5.2,3.5,1.5,0.2,Iris-setosa  Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
6.7,3.0,5.0,1.7,Iris-versicolor  Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
6.1,3.0,4.9,1.8,Iris-virginica  Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 1ms
5.2,3.4,1.4,0.2,Iris-setosa  Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
6.0,2.9,4.5,1.5,Iris-versicolor  Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
6.4,2.8,5.6,2.1,Iris-virginica  Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
4.7,3.2,1.6,0.2,Iris-setosa  Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
5.7,2.6,3.5,1.0,Iris-versicolor  Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
7.2,3.0,5.8,1.6,Iris-virginica  Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms **disagreed
4.8,3.1,1.6,0.2,Iris-setosa  Classifier1 (MultilayerPerceptron) 1ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
5.5,2.4,3.8,1.1,Iris-versicolor  Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
7.4,2.8,6.1,1.9,Iris-virginica  Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
5.4,3.4,1.5,0.4,Iris-setosa  Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
5.5,2.4,3.7,1.0,Iris-versicolor  Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
```

```
7.9,3.8,6.4,2.0,Iris-virginica                Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
5.2,4.1,1.5,0.1,Iris-setosa  Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
5.8,2.7,3.9,1.2,Iris-versicolor               Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
6.4,2.8,5.6,2.2,Iris-virginica                Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
5.5,4.2,1.4,0.2,Iris-setosa  Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
6.0,2.7,5.1,1.6,Iris-versicolor               Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms **disagreed
6.3,2.8,5.1,1.5,Iris-virginica                Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms **disagreed
4.9,3.1,1.5,0.1,Iris-setosa  Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 1ms
5.4,3.0,4.5,1.5,Iris-versicolor               Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
6.1,2.6,5.6,1.4,Iris-virginica                Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
5.0,3.2,1.2,0.2,Iris-setosa  Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
6.0,3.4,4.5,1.6,Iris-versicolor               Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
7.7,3.0,6.1,2.3,Iris-virginica                Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
5.5,3.5,1.3,0.2,Iris-setosa  Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
6.7,3.1,4.7,1.5,Iris-versicolor               Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
6.3,3.4,5.6,2.4,Iris-virginica                Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
4.9,3.1,1.5,0.1,Iris-setosa  Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
6.3,2.3,4.4,1.3,Iris-versicolor               Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
6.4,3.1,5.5,1.8,Iris-virginica                Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
4.4,3.0,1.3,0.2,Iris-setosa  Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
5.6,3.0,4.1,1.3,Iris-versicolor               Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
6.0,3.0,4.8,1.8,Iris-virginica                Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms **disagreed
5.1,3.4,1.5,0.2,Iris-setosa  Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
5.5,2.5,4.0,1.3,Iris-versicolor               Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
6.9,3.1,5.4,2.1,Iris-virginica                Classifier1 (MultilayerPerceptron) 13ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 1ms
5.0,3.5,1.3,0.3,Iris-setosa  Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
5.5,2.6,4.4,1.2,Iris-versicolor               Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
6.7,3.1,5.6,2.4,Iris-virginica                Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
4.5,2.3,1.3,0.3,Iris-setosa  Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 1ms
6.1,3.0,4.6,1.4,Iris-versicolor               Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
6.9,3.1,5.1,2.3,Iris-virginica                Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
4.4,3.2,1.3,0.2,Iris-setosa  Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
5.8,2.6,4.0,1.2,Iris-versicolor               Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
5.8,2.7,5.1,1.9,Iris-virginica                Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 1ms
5.0,3.5,1.6,0.6,Iris-setosa  Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 1ms
5.0,2.3,3.3,1.0,Iris-versicolor               Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
6.8,3.2,5.9,2.3,Iris-virginica                Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
5.1,3.8,1.9,0.4,Iris-setosa  Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
5.6,2.7,4.2,1.3,Iris-versicolor               Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
6.7,3.3,5.7,2.5,Iris-virginica                Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
4.8,3.0,1.4,0.3,Iris-setosa  Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
5.7,3.0,4.2,1.2,Iris-versicolor               Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
6.7,3.0,5.2,2.3,Iris-virginica                Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
5.1,3.8,1.6,0.2,Iris-setosa  Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 1ms
5.7,2.9,4.2,1.3,Iris-versicolor               Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
6.3,2.5,5.0,1.9,Iris-virginica                Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
4.6,3.2,1.4,0.2,Iris-setosa  Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
6.2,2.9,4.3,1.3,Iris-versicolor               Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
6.5,3.0,5.2,2.0,Iris-virginica                Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
5.3,3.7,1.5,0.2,Iris-setosa  Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
5.1,2.5,3.0,1.1,Iris-versicolor               Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
6.2,3.4,5.4,2.3,Iris-virginica                Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
5.0,3.3,1.4,0.2,Iris-setosa  Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
5.7,2.8,4.1,1.3,Iris-versicolor               Classifier1 (MultilayerPerceptron) 0ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms

*******************************************************************************
Image: Iris-setosa associated value: 0.0
Image: Iris-versicolor associated value: 1.0
Image: Iris-virginica associated value: 2.0
*******************************************************************************
Duration 83930ms
```

En aquest cas les discrepàncies son menors que amb l'utilització del IBk, per tant podem veure que podem arribar a jugar molt amb la definició de diferents agents, amb diferents classificadors i amb un conjunt de dades diferent.

6.2 Test 2. 5 Agents Classificadors

Utilitzant el mateix conjunt de dades que en el test anterior, es defineixen les següents propietats:

50% com a conjunt d'entrenament i la resta es passarà per el classificador. Es configurarà el sistema de la següent manera i amb els següents valors:

Manager:

```
#Manager Properties
#Location of the data
ArffDataSetLocation=C:/workspace/TFG-DistributedDecisionJade/data/iris.arff
#Percentage of the data dedicated to training
PercentageTrainingData=50
NumberOfClassifiers=5
#Classifier to use J48, IBk, MLP. Put "random" so the classifier can choose one randomly
ClassifierModule1=J48
ClassifierModule2=IBk
ClassifierModule3=MLP
ClassifierModule4=J48
ClassifierModule5=IBk
#Classifier uses the same Jade container true/false
ClassifiersUseSameContainer=false
#Output file after the decision has been made
OutputFileLocation=C:/workspace/TFG-DistributedDecisionJade/src/ddm/Output/iris_Diagnosis.out
#Verbosity true/false if we want the tool to add extra information on the screen
Verbosity=true
#Logging true/false to enable logging using log4j
Logging=true
```

Classifier:

```
#Classifier Properties
ApplicationPath=C:/workspace/TFG-DistributedDecisionJade/bin/
#Minimum and Maximum percentage of the data to use as a training
MinimumPercentageTraining=70
MaximumPercentageTraining=95
#Verbosity true/false if we want the tool to add extra information on the screen
Verbosity=true
#Logging true/false to enable logging using log4j
Logging=true
```

Definim que volem el entre el 70 i 95% de les dades destinades a l'entrenament. Per tant el classificador definirà un valor aleatori entre aquest rang.

L'execució de l'aplicació és la següent:

```
Classificador 1: J48, 93% conjunt d'entrenament.
Classificador 2: IBk, 85% conjunt d'entrenament.
Classificador 3: MLP, 71% conjunt d'entrenament.
Classificador 4: J48, 74% conjunt d'entrenament.
Classificador 5: IBk, 78% conjunt d'entrenament.

Name: Classifier2 Type: weka.classifiers.lazy.IBk Duration: 111ms
Name: Classifier5 Type: weka.classifiers.lazy.IBk Duration: 120ms
Name: Classifier4 Type: weka.classifiers.trees.J48 Duration: 128ms
Name: Classifier1 Type: weka.classifiers.trees.J48 Duration: 134ms
Name: Classifier3 Type: weka.classifiers.functions.MultilayerPerceptron Duration: 380ms
```

Amb els resultats següents:

```
*******************************************************************************
This file has been auto generated by DDM Distributed Decision Making System
*******************************************************************************

5.0,3.0,1.6,0.2,Iris-setosa
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-setosa(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-setosa(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-setosa(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-setosa(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-setosa(76.00%) 0ms
                Final Decision using OWA (Iris-setosa) value:0.00 in 3ms
6.6,3.0,4.4,1.4,Iris-versicolor
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-versicolor(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(92.00%) 1ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-versicolor(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-versicolor(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(76.00%) 1ms
                Final Decision using OWA (Iris-versicolor) value:1.00 in 1ms
7.2,3.2,6.0,1.8,Iris-virginica
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-virginica(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-virginica(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-virginica(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-virginica(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-virginica(76.00%) 0ms
                Final Decision using OWA (Iris-virginica) value:2.00 in 1ms
5.0,3.4,1.6,0.4,Iris-setosa
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-setosa(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-setosa(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-setosa(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-setosa(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-setosa(76.00%) 0ms
                Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
6.8,2.8,4.8,1.4,Iris-versicolor
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-versicolor(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(92.00%) 1ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-versicolor(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-versicolor(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(76.00%) 0ms
                Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
6.2,2.8,4.8,1.8,Iris-virginica
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-versicolor(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-virginica(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-virginica(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-virginica(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-virginica(76.00%) 0ms
                Final Decision using OWA (Iris-virginica) value:1.79 in 1ms
```

```
5.2,3.5,1.5,0.2,Iris-setosa
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-setosa(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-setosa(92.00%) 1ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-setosa(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-setosa(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-setosa(76.00%) 0ms
                Final Decision using OWA (Iris-setosa) value:0.00 in 1ms
6.7,3.0,5.0,1.7,Iris-versicolor
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-virginica(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-virginica(92.00%) 1ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-versicolor(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-virginica(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(76.00%) 0ms
                Final Decision using OWA (Iris-virginica) value:1.62 in 0ms **disagreed
6.1,3.0,4.9,1.8,Iris-virginica
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-versicolor(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-virginica(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-virginica(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-virginica(76.00%) 0ms
                Final Decision using OWA (Iris-virginica) value:1.55 in 0ms
5.2,3.4,1.4,0.2,Iris-setosa
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-setosa(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-setosa(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-setosa(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-setosa(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-setosa(76.00%) 0ms
                Final Decision using OWA (Iris-setosa) value:0.00 in 1ms
6.0,2.9,4.5,1.5,Iris-versicolor
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-versicolor(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-versicolor(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-versicolor(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(76.00%) 0ms
                Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
6.4,2.8,5.6,2.1,Iris-virginica
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-virginica(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-virginica(92.00%) 1ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-virginica(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-virginica(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-virginica(76.00%) 0ms
                Final Decision using OWA (Iris-virginica) value:2.00 in 1ms
4.7,3.2,1.6,0.2,Iris-setosa
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-setosa(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-setosa(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-setosa(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-setosa(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-setosa(76.00%) 1ms
                Final Decision using OWA (Iris-setosa) value:0.00 in 1ms
5.7,2.6,3.5,1.0,Iris-versicolor
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-versicolor(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-versicolor(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-versicolor(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(76.00%) 1ms
                Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
7.2,3.0,5.8,1.6,Iris-virginica
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-virginica(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-virginica(92.00%) 1ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-virginica(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-versicolor(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-virginica(76.00%) 0ms
                Final Decision using OWA (Iris-virginica) value:1.82 in 0ms
4.8,3.1,1.6,0.2,Iris-setosa
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-setosa(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-setosa(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-setosa(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-setosa(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-setosa(76.00%) 0ms
                Final Decision using OWA (Iris-setosa) value:0.00 in 1ms
5.5,2.4,3.8,1.1,Iris-versicolor
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-versicolor(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-versicolor(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-versicolor(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(76.00%) 0ms
                Final Decision using OWA (Iris-versicolor) value:1.00 in 1ms
7.4,2.8,6.1,1.9,Iris-virginica
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-virginica(84.00%) 1ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-virginica(92.00%) 1ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-virginica(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-virginica(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-virginica(76.00%) 0ms
                Final Decision using OWA (Iris-virginica) value:2.00 in 1ms
5.4,3.4,1.5,0.4,Iris-setosa
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-setosa(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-setosa(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-setosa(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-setosa(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-setosa(76.00%) 0ms
                Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
5.5,2.4,3.7,1.0,Iris-versicolor
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-versicolor(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(92.00%) 0ms
```

Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-versicolor(73.33%) 0ms
Classifier4 (weka.classifiers.trees.J48) decision:Iris-versicolor(69.33%) 0ms
Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(76.00%) 0ms
Final Decision using OWA (Iris-versicolor) value:1.00 in 1ms
7.9,3.8,6.4,2.0,Iris-virginica
Classifier1 (weka.classifiers.trees.J48) decision:Iris-virginica(84.00%) 0ms
Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-virginica(92.00%) 0ms
Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-virginica(73.33%) 0ms
Classifier4 (weka.classifiers.trees.J48) decision:Iris-virginica(69.33%) 0ms
Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-virginica(76.00%) 0ms
Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
5.2,4.1,1.5,0.1,Iris-setosa
Classifier1 (weka.classifiers.trees.J48) decision:Iris-setosa(84.00%) 0ms
Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-setosa(92.00%) 0ms
Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-setosa(73.33%) 1ms
Classifier4 (weka.classifiers.trees.J48) decision:Iris-setosa(69.33%) 0ms
Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-setosa(76.00%) 0ms
Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
5.8,2.7,3.9,1.2,Iris-versicolor
Classifier1 (weka.classifiers.trees.J48) decision:Iris-versicolor(84.00%) 0ms
Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(92.00%) 0ms
Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-versicolor(73.33%) 0ms
Classifier4 (weka.classifiers.trees.J48) decision:Iris-versicolor(69.33%) 0ms
Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(76.00%) 0ms
Final Decision using OWA (Iris-versicolor) value:1.00 in 1ms
6.4,2.8,5.6,2.2,Iris-virginica
Classifier1 (weka.classifiers.trees.J48) decision:Iris-virginica(84.00%) 0ms
Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-virginica(92.00%) 0ms
Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-virginica(73.33%) 0ms
Classifier4 (weka.classifiers.trees.J48) decision:Iris-virginica(69.33%) 0ms
Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-virginica(76.00%) 1ms
Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
5.5,4.2,1.4,0.2,Iris-setosa
Classifier1 (weka.classifiers.trees.J48) decision:Iris-setosa(84.00%) 0ms
Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-setosa(92.00%) 0ms
Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-setosa(73.33%) 0ms
Classifier4 (weka.classifiers.trees.J48) decision:Iris-setosa(69.33%) 0ms
Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-setosa(76.00%) 0ms
Final Decision using OWA (Iris-setosa) value:0.00 in 1ms
6.0,2.7,5.1,1.6,Iris-versicolor
Classifier1 (weka.classifiers.trees.J48) decision:Iris-virginica(84.00%) 0ms
Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(92.00%) 0ms
Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-virginica(73.33%) 0ms
Classifier4 (weka.classifiers.trees.J48) decision:Iris-versicolor(69.33%) 0ms
Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-virginica(76.00%) 0ms
Final Decision using OWA (Iris-virginica) value:1.59 in 1ms **disagreed
6.3,2.8,5.1,1.5,Iris-virginica
Classifier1 (weka.classifiers.trees.J48) decision:Iris-virginica(84.00%) 0ms
Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(92.00%) 0ms
Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-versicolor(73.33%) 0ms
Classifier4 (weka.classifiers.trees.J48) decision:Iris-versicolor(69.33%) 0ms
Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(76.00%) 1ms
Final Decision using OWA (Iris-versicolor) value:1.21 in 1ms **disagreed
4.9,3.1,1.5,0.1,Iris-setosa
Classifier1 (weka.classifiers.trees.J48) decision:Iris-setosa(84.00%) 0ms
Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-setosa(92.00%) 0ms
Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-setosa(73.33%) 0ms
Classifier4 (weka.classifiers.trees.J48) decision:Iris-setosa(69.33%) 0ms
Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-setosa(76.00%) 0ms
Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
5.4,3.0,4.5,1.5,Iris-versicolor
Classifier1 (weka.classifiers.trees.J48) decision:Iris-versicolor(84.00%) 0ms
Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(92.00%) 0ms
Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-versicolor(73.33%) 0ms
Classifier4 (weka.classifiers.trees.J48) decision:Iris-versicolor(69.33%) 0ms
Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(76.00%) 0ms
Final Decision using OWA (Iris-versicolor) value:1.00 in 1ms
6.1,2.6,5.6,1.4,Iris-virginica
Classifier1 (weka.classifiers.trees.J48) decision:Iris-virginica(84.00%) 0ms
Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(92.00%) 0ms
Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-virginica(73.33%) 0ms
Classifier4 (weka.classifiers.trees.J48) decision:Iris-versicolor(69.33%) 0ms
Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(76.00%) 0ms
Final Decision using OWA (Iris-versicolor) value:1.40 in 1ms **disagreed
5.0,3.2,1.2,0.2,Iris-setosa
Classifier1 (weka.classifiers.trees.J48) decision:Iris-setosa(84.00%) 0ms
Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-setosa(92.00%) 0ms
Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-setosa(73.33%) 0ms
Classifier4 (weka.classifiers.trees.J48) decision:Iris-setosa(69.33%) 0ms
Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-setosa(76.00%) 0ms
Final Decision using OWA (Iris-setosa) value:0.00 in 1ms
6.0,3.4,4.5,1.6,Iris-versicolor
Classifier1 (weka.classifiers.trees.J48) decision:Iris-versicolor(84.00%) 0ms
Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(92.00%) 0ms
Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-versicolor(73.33%) 0ms
Classifier4 (weka.classifiers.trees.J48) decision:Iris-versicolor(69.33%) 0ms
Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(76.00%) 0ms
Final Decision using OWA (Iris-versicolor) value:1.00 in 1ms
7.7,3.0,6.1,2.3,Iris-virginica
Classifier1 (weka.classifiers.trees.J48) decision:Iris-virginica(84.00%) 0ms
Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-virginica(92.00%) 0ms
Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-virginica(73.33%) 0ms
Classifier4 (weka.classifiers.trees.J48) decision:Iris-virginica(69.33%) 0ms
Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-virginica(76.00%) 0ms

```
                Final Decision using OWA (Iris-virginica) value:2.00 in 1ms
5.5,3.5,1.3,0.2,Iris-setosa
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-setosa(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-setosa(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-setosa(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-setosa(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-setosa(76.00%) 0ms
                Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
6.7,3.1,4.7,1.5,Iris-versicolor
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-versicolor(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-versicolor(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-versicolor(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(76.00%) 0ms
                Final Decision using OWA (Iris-versicolor) value:1.00 in 1ms
6.3,3.4,5.6,2.4,Iris-virginica
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-virginica(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-virginica(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-virginica(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-virginica(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-virginica(76.00%) 0ms
                Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
4.9,3.1,1.5,0.1,Iris-setosa
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-setosa(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-setosa(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-setosa(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-setosa(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-setosa(76.00%) 0ms
                Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
6.3,2.3,4.4,1.3,Iris-versicolor
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-versicolor(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-versicolor(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-versicolor(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(76.00%) 0ms
                Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
6.4,3.1,5.5,1.8,Iris-virginica
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-virginica(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-virginica(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-virginica(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-virginica(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-virginica(76.00%) 0ms
                Final Decision using OWA (Iris-virginica) value:2.00 in 1ms
4.4,3.0,1.3,0.2,Iris-setosa
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-setosa(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-setosa(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-setosa(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-setosa(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-setosa(76.00%) 0ms
                Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
5.6,3.0,4.1,1.3,Iris-versicolor
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-versicolor(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-versicolor(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-versicolor(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(76.00%) 0ms
                Final Decision using OWA (Iris-versicolor) value:1.00 in 1ms
6.0,3.0,4.8,1.8,Iris-virginica
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-versicolor(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-virginica(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-virginica(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-virginica(76.00%) 0ms
                Final Decision using OWA (Iris-virginica) value:1.55 in 0ms
5.1,3.4,1.5,0.2,Iris-setosa
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-setosa(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-setosa(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-setosa(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-setosa(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-setosa(76.00%) 0ms
                Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
5.5,2.5,4.0,1.3,Iris-versicolor
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-versicolor(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-versicolor(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-versicolor(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(76.00%) 0ms
                Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
6.9,3.1,5.4,2.1,Iris-virginica
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-virginica(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-virginica(92.00%) 1ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-virginica(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-virginica(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-virginica(76.00%) 0ms
                Final Decision using OWA (Iris-virginica) value:2.00 in 1ms
5.0,3.5,1.3,0.3,Iris-setosa
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-setosa(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-setosa(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-setosa(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-setosa(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-setosa(76.00%) 0ms
                Final Decision using OWA (Iris-setosa) value:0.00 in 1ms
5.5,2.6,4.4,1.2,Iris-versicolor
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-versicolor(84.00%) 0ms
```

```
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-versicolor(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-versicolor(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(76.00%) 0ms
                Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
6.7,3.1,5.6,2.4,Iris-virginica
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-virginica(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-virginica(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-virginica(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-virginica(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-virginica(76.00%) 0ms
                Final Decision using OWA (Iris-virginica) value:2.00 in 1ms
4.5,2.3,1.3,0.3,Iris-setosa
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-setosa(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-setosa(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-setosa(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-setosa(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-setosa(76.00%) 0ms
                Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
6.1,3.0,4.6,1.4,Iris-versicolor
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-versicolor(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-versicolor(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-versicolor(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(76.00%) 1ms
                Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
6.9,3.1,5.1,2.3,Iris-virginica
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-virginica(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-virginica(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-virginica(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-virginica(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-virginica(76.00%) 0ms
                Final Decision using OWA (Iris-virginica) value:2.00 in 1ms
4.4,3.2,1.3,0.2,Iris-setosa
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-setosa(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-setosa(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-setosa(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-setosa(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-setosa(76.00%) 0ms
                Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
5.8,2.6,4.0,1.2,Iris-versicolor
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-versicolor(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-versicolor(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-versicolor(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(76.00%) 0ms
                Final Decision using OWA (Iris-versicolor) value:1.00 in 1ms
5.8,2.7,5.1,1.9,Iris-virginica
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-virginica(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-virginica(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-virginica(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-virginica(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-virginica(76.00%) 0ms
                Final Decision using OWA (Iris-virginica) value:2.00 in 1ms
5.0,3.5,1.6,0.6,Iris-setosa
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-versicolor(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-setosa(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-setosa(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-versicolor(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-setosa(76.00%) 0ms
                Final Decision using OWA (Iris-setosa) value:0.39 in 0ms
5.0,2.3,3.3,1.0,Iris-versicolor
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-versicolor(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-versicolor(73.33%) 1ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-versicolor(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(76.00%) 0ms
                Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
6.8,3.2,5.9,2.3,Iris-virginica
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-virginica(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-virginica(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-virginica(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-virginica(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-virginica(76.00%) 0ms
                Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
5.1,3.8,1.9,0.4,Iris-setosa
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-setosa(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-setosa(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-setosa(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-setosa(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-setosa(76.00%) 0ms
                Final Decision using OWA (Iris-setosa) value:0.00 in 1ms
5.6,2.7,4.2,1.3,Iris-versicolor
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-versicolor(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-versicolor(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-versicolor(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(76.00%) 0ms
                Final Decision using OWA (Iris-versicolor) value:1.00 in 1ms
6.7,3.3,5.7,2.5,Iris-virginica
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-virginica(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-virginica(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-virginica(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-virginica(69.33%) 0ms
```

```
		Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-virginica(76.00%) 0ms
			Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
4.8,3.0,1.4,0.3,Iris-setosa
		Classifier1 (weka.classifiers.trees.J48) decision:Iris-setosa(84.00%) 0ms
		Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-setosa(92.00%) 0ms
		Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-setosa(73.33%) 0ms
		Classifier4 (weka.classifiers.trees.J48) decision:Iris-setosa(69.33%) 0ms
		Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-setosa(76.00%) 0ms
			Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
5.7,3.0,4.2,1.2,Iris-versicolor
		Classifier1 (weka.classifiers.trees.J48) decision:Iris-versicolor(84.00%) 1ms
		Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(92.00%) 0ms
		Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-versicolor(73.33%) 0ms
		Classifier4 (weka.classifiers.trees.J48) decision:Iris-versicolor(69.33%) 0ms
		Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(76.00%) 0ms
			Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
6.7,3.0,5.2,2.3,Iris-virginica
		Classifier1 (weka.classifiers.trees.J48) decision:Iris-virginica(84.00%) 0ms
		Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-virginica(92.00%) 0ms
		Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-virginica(73.33%) 0ms
		Classifier4 (weka.classifiers.trees.J48) decision:Iris-virginica(69.33%) 0ms
		Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-virginica(76.00%) 0ms
			Final Decision using OWA (Iris-virginica) value:2.00 in 1ms
5.1,3.8,1.6,0.2,Iris-setosa
		Classifier1 (weka.classifiers.trees.J48) decision:Iris-setosa(84.00%) 0ms
		Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-setosa(92.00%) 0ms
		Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-setosa(73.33%) 0ms
		Classifier4 (weka.classifiers.trees.J48) decision:Iris-setosa(69.33%) 0ms
		Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-setosa(76.00%) 0ms
			Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
5.7,2.9,4.2,1.3,Iris-versicolor
		Classifier1 (weka.classifiers.trees.J48) decision:Iris-versicolor(84.00%) 0ms
		Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(92.00%) 0ms
		Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-versicolor(73.33%) 1ms
		Classifier4 (weka.classifiers.trees.J48) decision:Iris-versicolor(69.33%) 0ms
		Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(76.00%) 0ms
			Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
6.3,2.5,5.0,1.9,Iris-virginica
		Classifier1 (weka.classifiers.trees.J48) decision:Iris-virginica(84.00%) 0ms
		Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-virginica(92.00%) 0ms
		Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-virginica(73.33%) 0ms
		Classifier4 (weka.classifiers.trees.J48) decision:Iris-virginica(69.33%) 0ms
		Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-virginica(76.00%) 0ms
			Final Decision using OWA (Iris-virginica) value:2.00 in 1ms
4.6,3.2,1.4,0.2,Iris-setosa
		Classifier1 (weka.classifiers.trees.J48) decision:Iris-setosa(84.00%) 0ms
		Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-setosa(92.00%) 0ms
		Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-setosa(73.33%) 0ms
		Classifier4 (weka.classifiers.trees.J48) decision:Iris-setosa(69.33%) 0ms
		Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-setosa(76.00%) 0ms
			Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
6.2,2.9,4.3,1.3,Iris-versicolor
		Classifier1 (weka.classifiers.trees.J48) decision:Iris-versicolor(84.00%) 0ms
		Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(92.00%) 0ms
		Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-versicolor(73.33%) 0ms
		Classifier4 (weka.classifiers.trees.J48) decision:Iris-versicolor(69.33%) 0ms
		Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(76.00%) 0ms
			Final Decision using OWA (Iris-versicolor) value:1.00 in 1ms
6.5,3.0,5.2,2.0,Iris-virginica
		Classifier1 (weka.classifiers.trees.J48) decision:Iris-virginica(84.00%) 0ms
		Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-virginica(92.00%) 0ms
		Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-virginica(73.33%) 0ms
		Classifier4 (weka.classifiers.trees.J48) decision:Iris-virginica(69.33%) 0ms
		Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-virginica(76.00%) 0ms
			Final Decision using OWA (Iris-virginica) value:2.00 in 0ms
5.3,3.7,1.5,0.2,Iris-setosa
		Classifier1 (weka.classifiers.trees.J48) decision:Iris-setosa(84.00%) 0ms
		Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-setosa(92.00%) 0ms
		Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-setosa(73.33%) 0ms
		Classifier4 (weka.classifiers.trees.J48) decision:Iris-setosa(69.33%) 0ms
		Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-setosa(76.00%) 0ms
			Final Decision using OWA (Iris-setosa) value:0.00 in 2ms
5.1,2.5,3.0,1.1,Iris-versicolor
		Classifier1 (weka.classifiers.trees.J48) decision:Iris-versicolor(84.00%) 0ms
		Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(92.00%) 0ms
		Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-versicolor(73.33%) 0ms
		Classifier4 (weka.classifiers.trees.J48) decision:Iris-versicolor(69.33%) 0ms
		Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(76.00%) 0ms
			Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms
6.2,3.4,5.4,2.3,Iris-virginica
		Classifier1 (weka.classifiers.trees.J48) decision:Iris-virginica(84.00%) 0ms
		Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-virginica(92.00%) 0ms
		Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-virginica(73.33%) 0ms
		Classifier4 (weka.classifiers.trees.J48) decision:Iris-virginica(69.33%) 0ms
		Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-virginica(76.00%) 0ms
			Final Decision using OWA (Iris-virginica) value:2.00 in 1ms
5.0,3.3,1.4,0.2,Iris-setosa
		Classifier1 (weka.classifiers.trees.J48) decision:Iris-setosa(84.00%) 0ms
		Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-setosa(92.00%) 0ms
		Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-setosa(73.33%) 0ms
		Classifier4 (weka.classifiers.trees.J48) decision:Iris-setosa(69.33%) 0ms
		Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-setosa(76.00%) 0ms
			Final Decision using OWA (Iris-setosa) value:0.00 in 0ms
5.7,2.8,4.1,1.3,Iris-versicolor
```

```
		Classifier1 (weka.classifiers.trees.J48) decision:Iris-versicolor(84.00%) 0ms
		Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(92.00%) 0ms
		Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-versicolor(73.33%) 0ms
		Classifier4 (weka.classifiers.trees.J48) decision:Iris-versicolor(69.33%) 0ms
		Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(76.00%) 0ms
				Final Decision using OWA (Iris-versicolor) value:1.00 in 1ms

********************************************************************************
Image: Iris-setosa associated value: 0.0
Image: Iris-versicolor associated value: 1.0
Image: Iris-virginica associated value: 2.0
********************************************************************************
Duration 124245ms
```

Apareixen 4 entrades amb discrepància sobre la classificació original.

```
6.1,2.6,5.6,1.4,Iris-virginica
		Classifier1 (weka.classifiers.trees.J48) decision:Iris-virginica(84.00%) 0ms
		Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(92.00%) 0ms
		Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-virginica(73.33%) 0ms
		Classifier4 (weka.classifiers.trees.J48) decision:Iris-versicolor(69.33%) 0ms
		Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(76.00%) 0ms
				Final Decision using OWA (Iris-versicolor) value:1.40 in 1ms **disagreed
```

La classificació inicial indica que és Iris-virginica, però tenim dues opinions diferents. 2 agents corroboren aquesta classificació i els altres 3 indiquen que és Iris-versicolor. El valor final és 1.40, per tant està a 0.40 de 1 i a 0.60 de 2, per tant la decisió final és Iris-versicolor.

Les següents figures mostren les comunicacions entre agents i la seva creació:

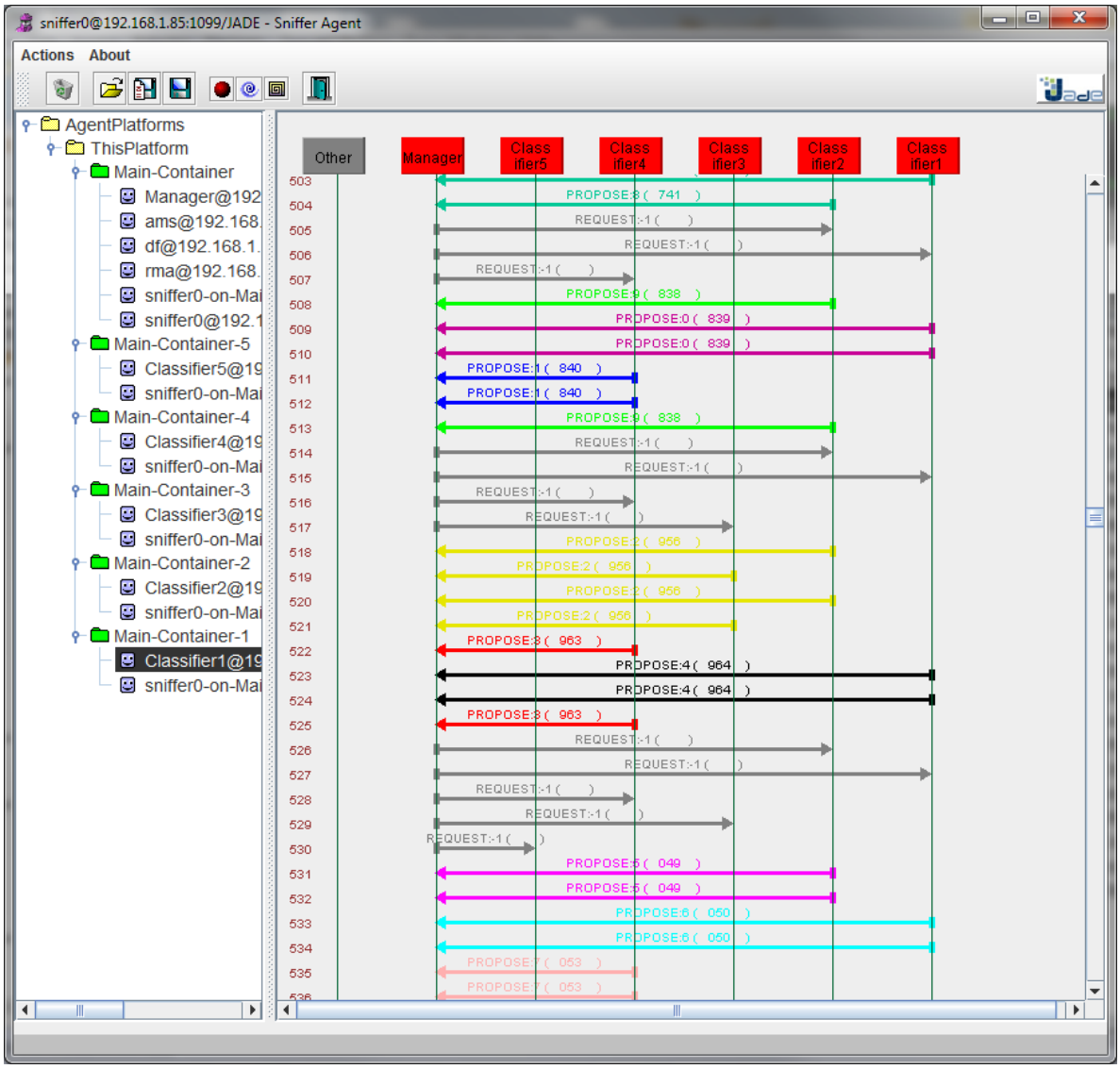


Figura 29 Exemple comunicació entre agents.

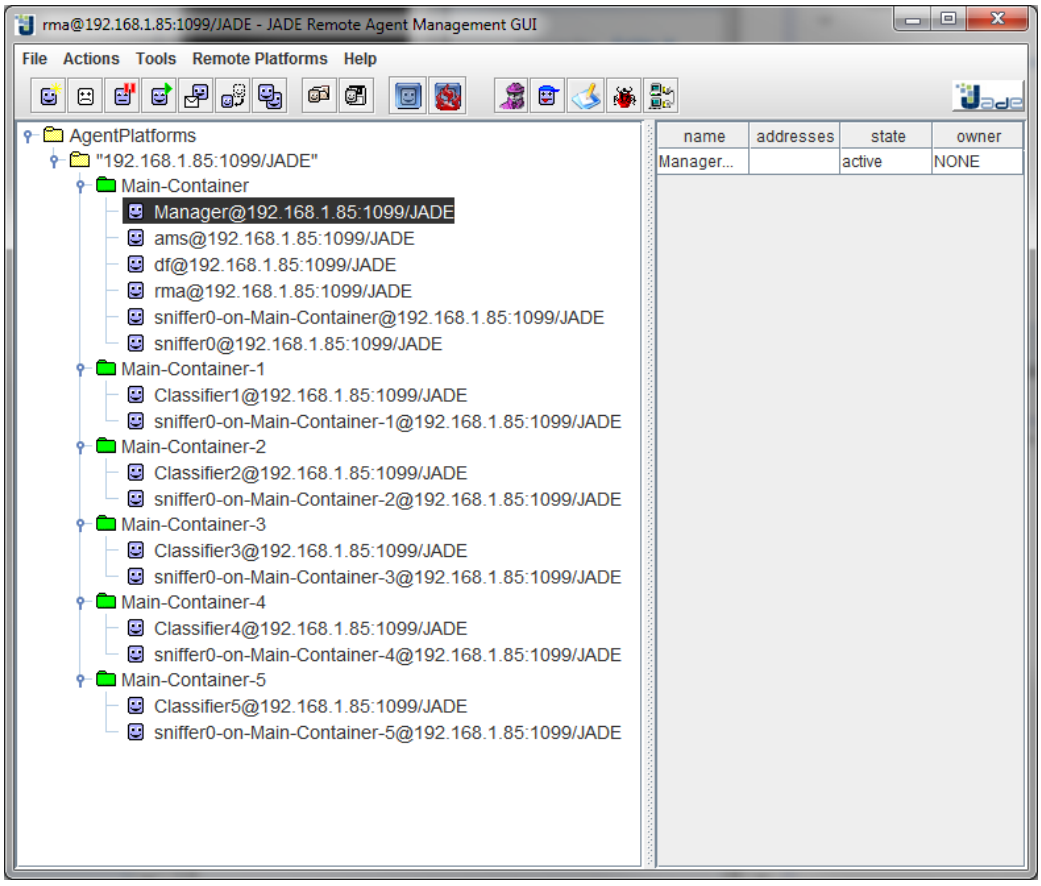


Figura 30 Exemple creació dels agents en diferents contenidors

7 Manual d'instal·lació de la solució

El codi font es pot trobar al següent repositori online:

- **https://github.com/JordiCorbilla/DDM**

Cal descarregar la versió **master** utilitzant el botó de descàrrega "Download ZIP".

Un cop descarregat caldrà configurar el CLASSPATH del projecte:

```
%JADE_HOME%\lib\jade.jar;
%JADE_HOME%\lib\common-codec\common-codec-1.3.jar;
%JADE_HOME%\classes;
C:\DDM\lib\weka-stable-3.6.6.jar;
C:\DDM\lib\log4j-1.2.1.7.jar;
C:\DDM\bin\ddm\agents;
```

Caldrà doncs definir la ruta dels agents a executar, la ruta de la localització del jade.jar, el common-code-1.3.jar, el weka (weka-stable-3.6.6.jar) i el log4j (log4j-1.2.1.7.jar).

Cal carregar el projecte a l'Eclipse per a poder compilar i generar els fitxers .class que son els que executarem:

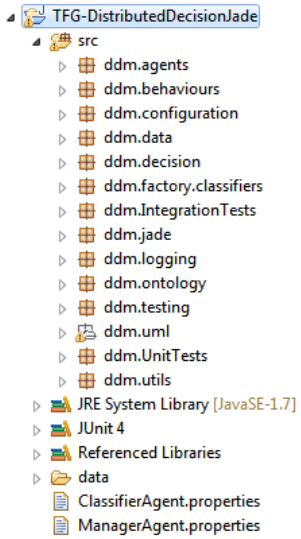


Figura 31 Estructura del projecte

Un cop configurat correctament i tinguem les classes correctament construïdes (fitxers .class), podrem passar a l'execució de l'aplicació.

8 Execució de la solució

Per a la correcta execució de la solució cal primer realitzar la configuració bàsica del sistema ja que s'han de referenciar les llibreries utilitzades en les variables d'entorn. L'execució s'ha realitzat sota Windows 7 64bits.

Es disposa d'un fitxer de comandes que genera un Agent Manager per defecte.

runDDMManager.bat en la carpeta **\bin**

```
java jade.Boot -gui -local-port 1099 -agents Manager:ddm.agents.ManagerAgent
```

Aquest a l'iniciar-se ens mostra un menú per consola que ens permet realitzar l'execució del sistema:

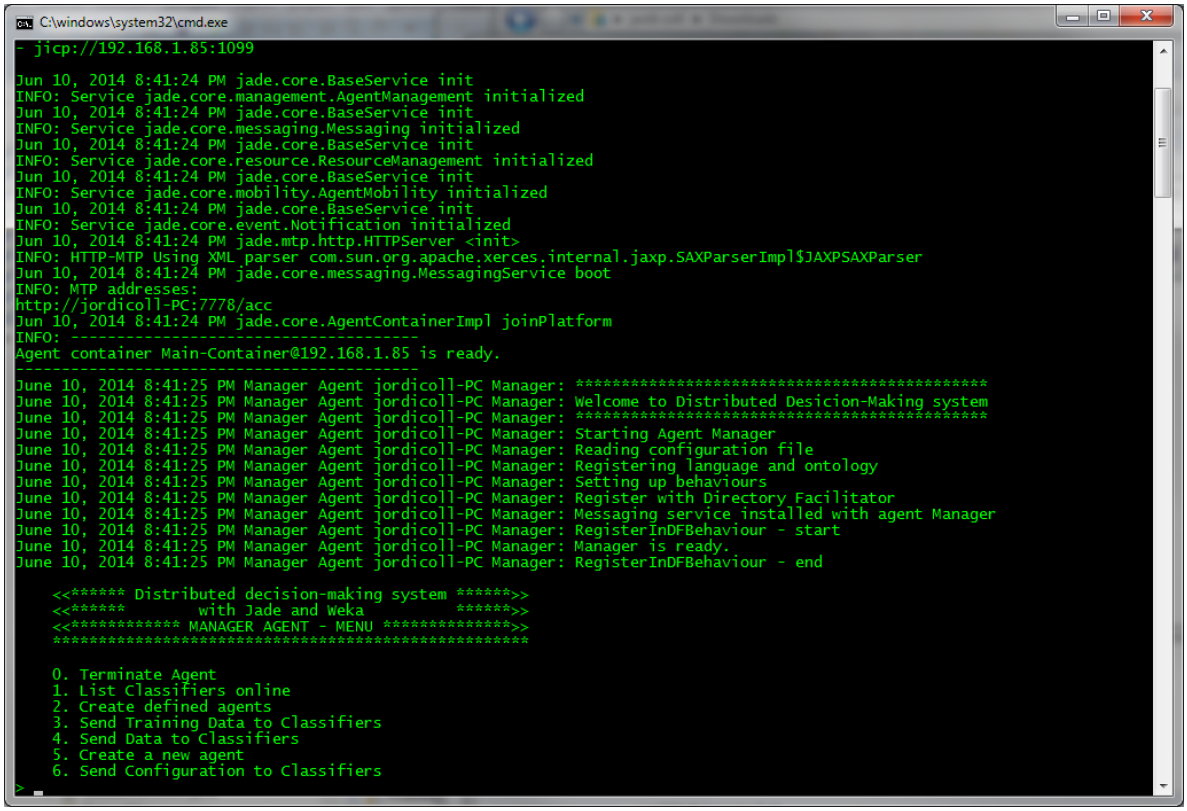


Figura 32 Exemple execució solució

Com a accions principals tenim:

- **0** - **Finalitzar execució de l'agent**. Bàsicament acaba amb l'execució de l'agent principal i dels classificadors.
- **1** - **Mostra classificadors en línia**. Aquesta comanda mostra la informació dels agents que estan en línia i amb els que podrà comunicar:

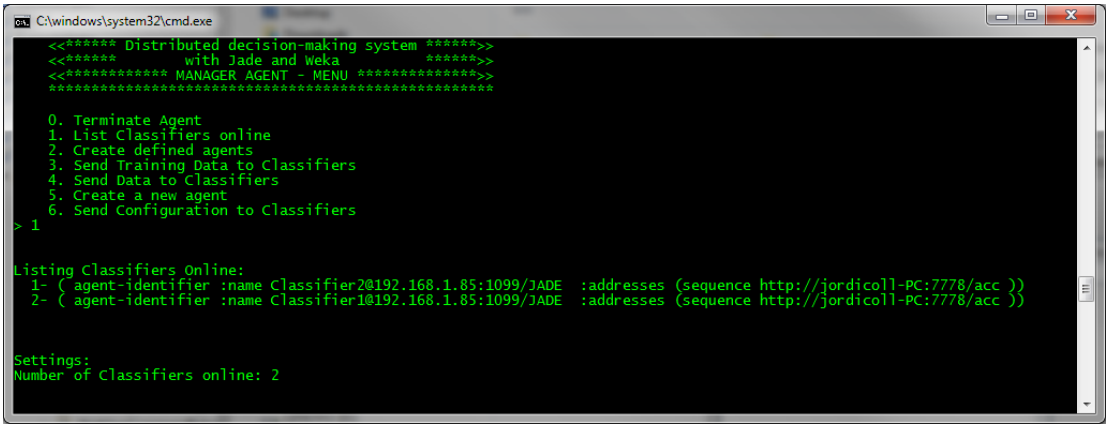


Figura 33 Mostra de classificadors en línia

Aquesta llista de classificadors és la que s'utilitzarà a l'hora d'enviar les dades d'entrenament i classificació.

- **2** - **Creació agents predefinits**. Aquesta opció crearà el número d'agents classificadors que tenim definits segons la propietat *NumberOfClassifiers* del Manager. Un cop s'han generat els classificadors, el manager enviarà la configuració a cadascú d'aquests i cada agent respondrà amb les dades que s'han imposat a cada un d'ells.

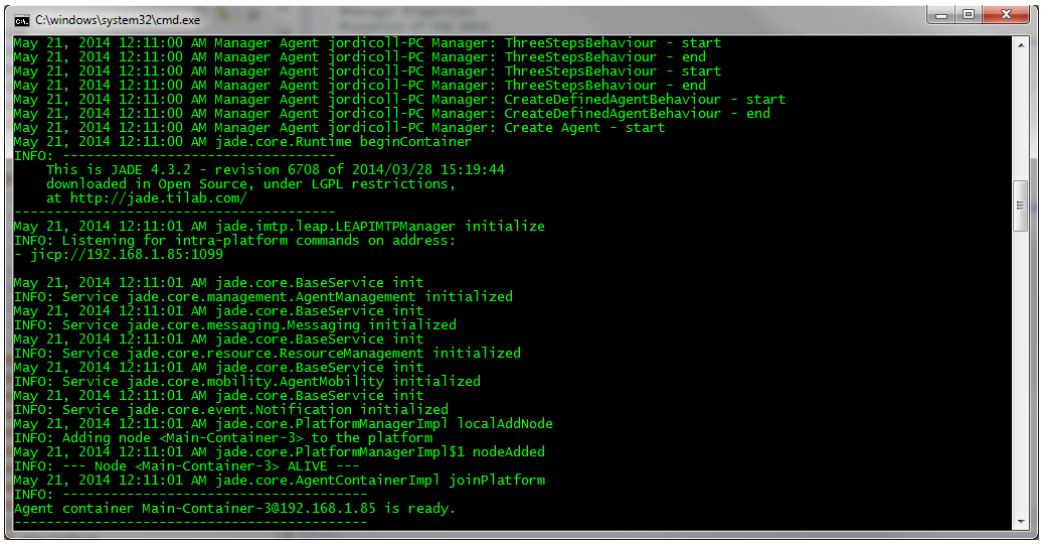


Figura 34 Creació d'un agent

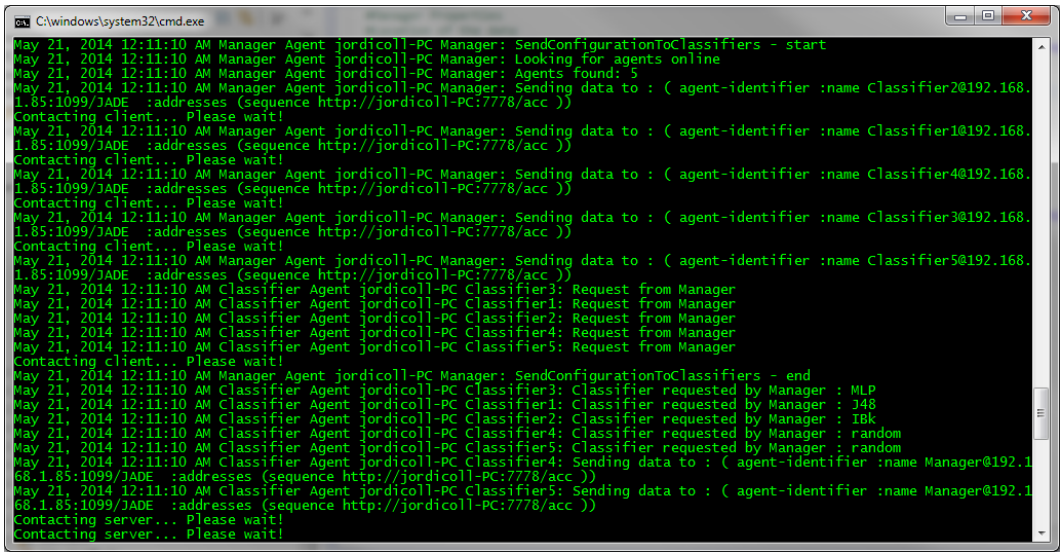


Figura 35 Enviament configuració i resposta dels classificadors

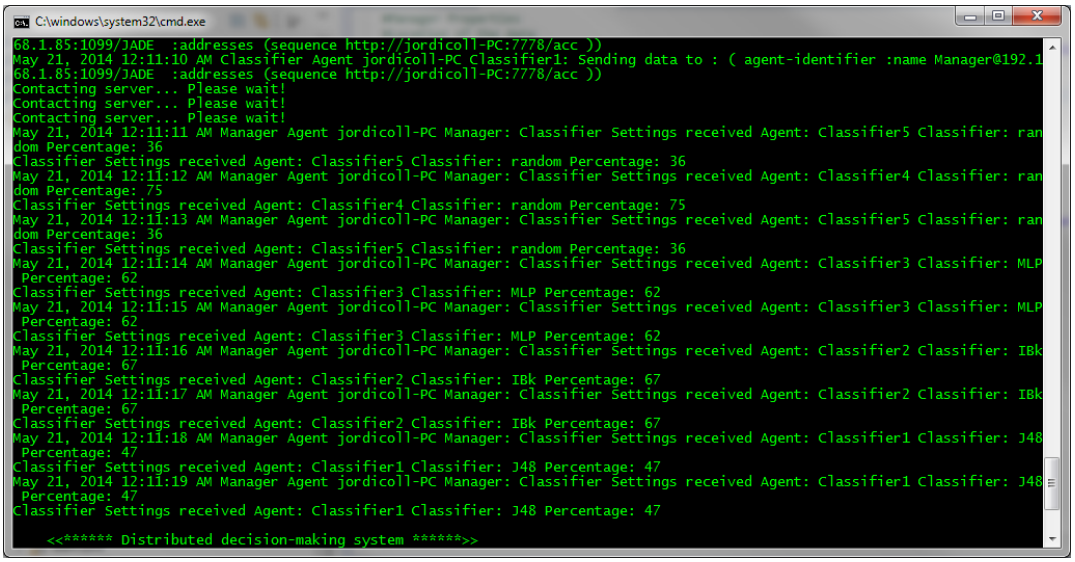


Figura 36 Resposta final de l'agent amb els paràmetres definits

- **3** - **Enviament del conjunt de dades d'entrenament**. En aquest apartat, disposem dels agents creats i sabem la configuració de cada agent, és a dir, sabem paràmetres bàsics com: l'algorisme que utilitzaran i quin percentatge de les dades destinades a l'entrenament volen. A partir d'aquí, aquesta acció realitzarà la divisió pertinent del conjunt de dades d'entrenament i enviarà aquests paquets a cada classificador. La resposta de cada agent és un ACK indicant que l'agent és entrenat correctament.

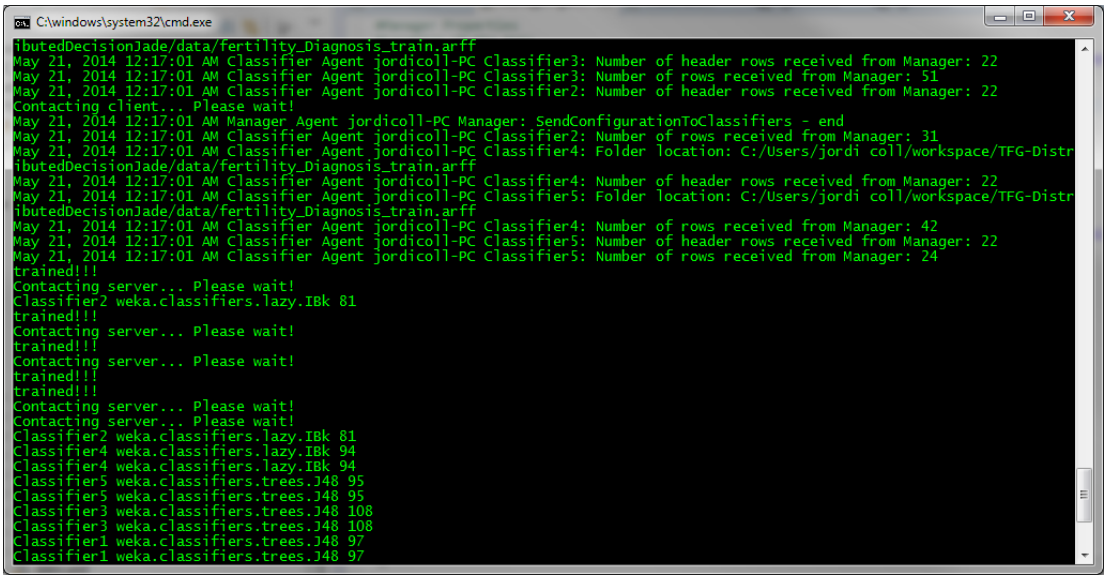


Figura 37 Enviament dades d'entrenament

- **4** - **Envia conjunt de dades a classificar**. Un cop disposem de tots els agents entrenats i en línia, aquests queden a l'espera per a l'enviament de dades a classificar. El protocol de comunicacions definit, realitza l'enviament de les dades a cada agent i després cada agent respon amb una predicció. L'agent manager composa totes aquestes respostes i pren una decisió que escriu en un fitxer de sortida.

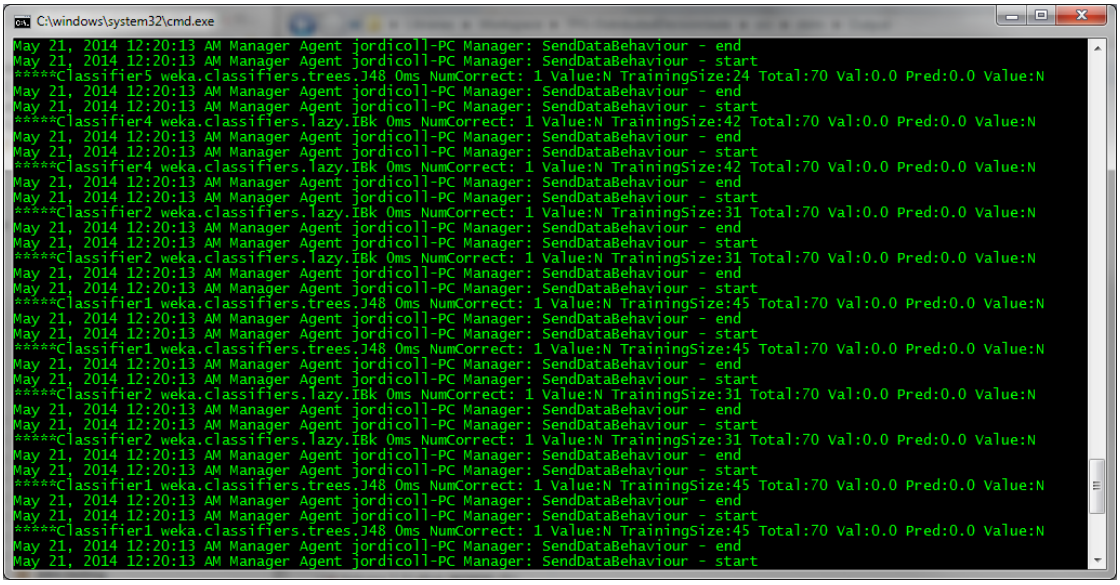


Figura 38 Exemple enviament de dades i resposta dels classificadors

- **5** - **Crear nous agents**. Aquesta opció permet crear agents al vol, és a dir, ens permet crear agents que funcionen sense haver-los de configurar. D'aquesta manera podem iniciar el nostre procés definint una sèrie d'agents i continuar afegint-ne més sense haver d'aturar l'aplicació.

- **6** - **Enviar la configuració als agents classificadors**. Aquesta opció ens permet enviar la configuració definida en les propietats per a que així cada agent tingui una configuració predefinida. Si no s'executa aquesta acció, els agents simplement trien una configuració aleatòria.

8.1 Mostra dels resultats

Un cop acabada l'execució de l'aplicació, les dades queden desades en un fitxer que s'ha definit prèviament en la secció de configuració de l'agent.

Si obrim aquest fitxer, podem veure les dades originals i llavors la decisió de cada classificador amb el seu pes de decisió i la decisió final intentada per a la seva millor visualització.

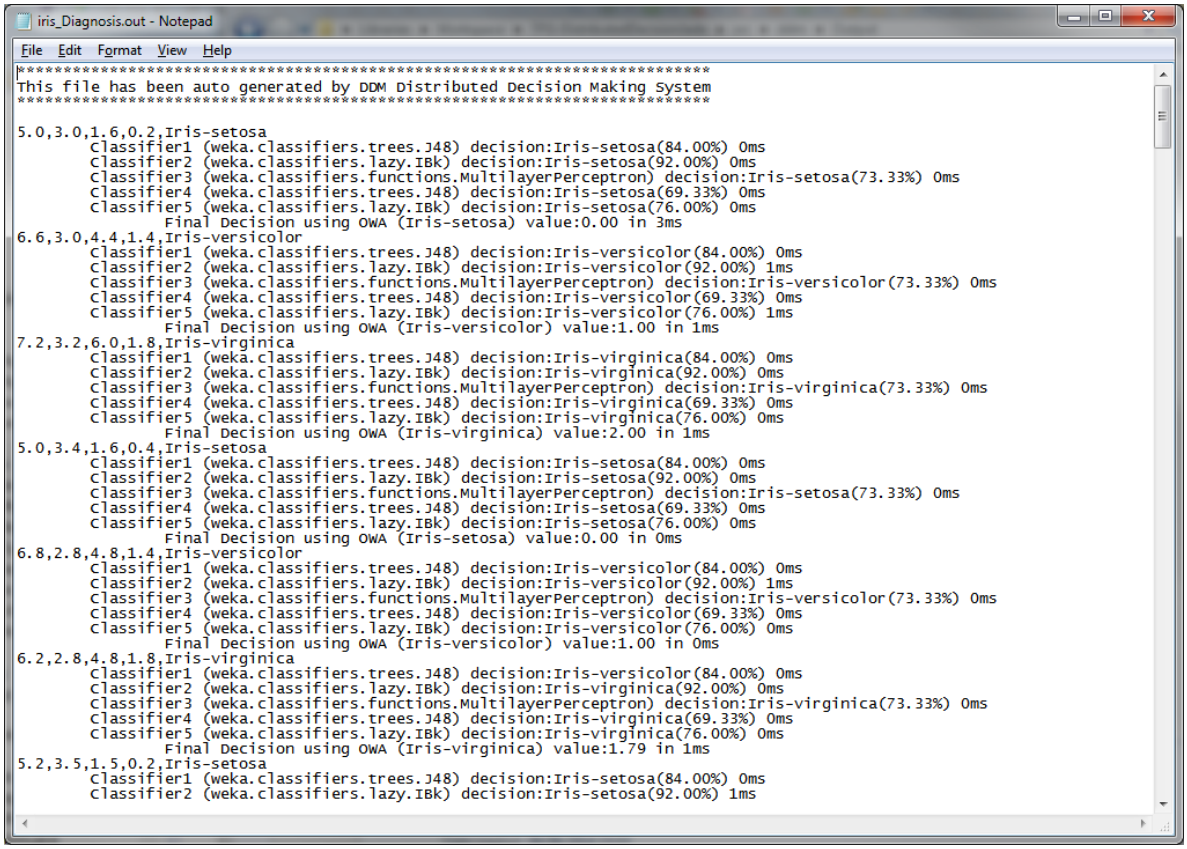


Figura 39 Fitxer resultant.

A més també es mostra el valor de les imatges generat i la duració total de l'execució del moment de la classificació:

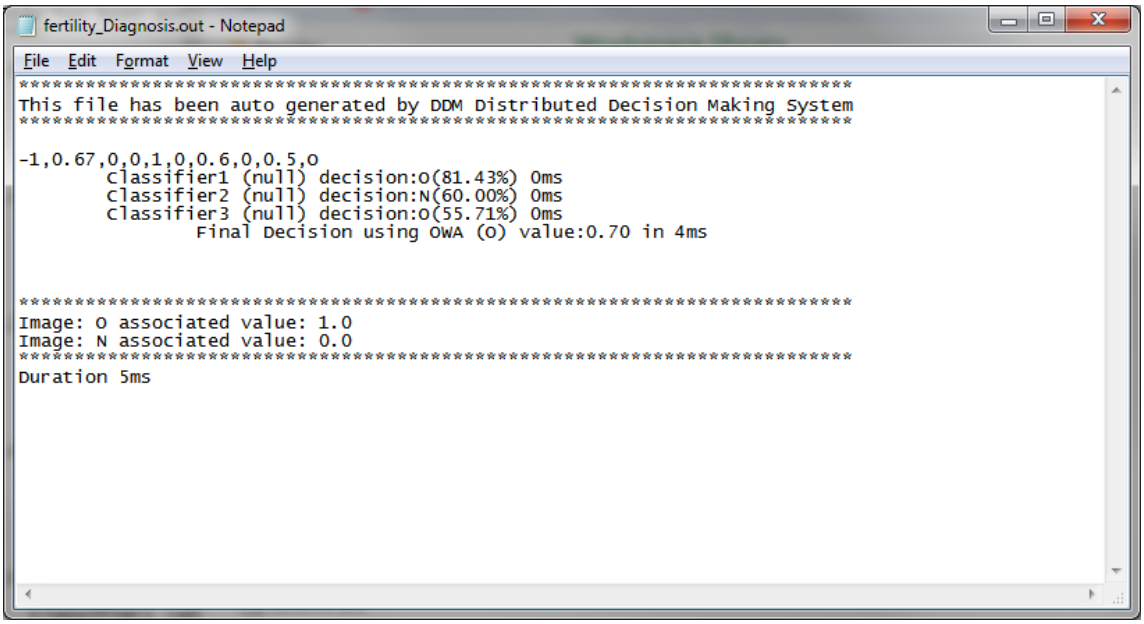


Figura 40 Fitxer resultant amb dades addicionals

8.2 Agent sniffer

Jade ens proporciona eines molt interessants per poder veure la comunicació que es realitza entre els agents instal·lant un *sniffer* i comprovar així que els REQUEST/INFORM o altres tipus estan correctament codificats i la informació es transmet sense problemes utilitzant l'ontologia definida:

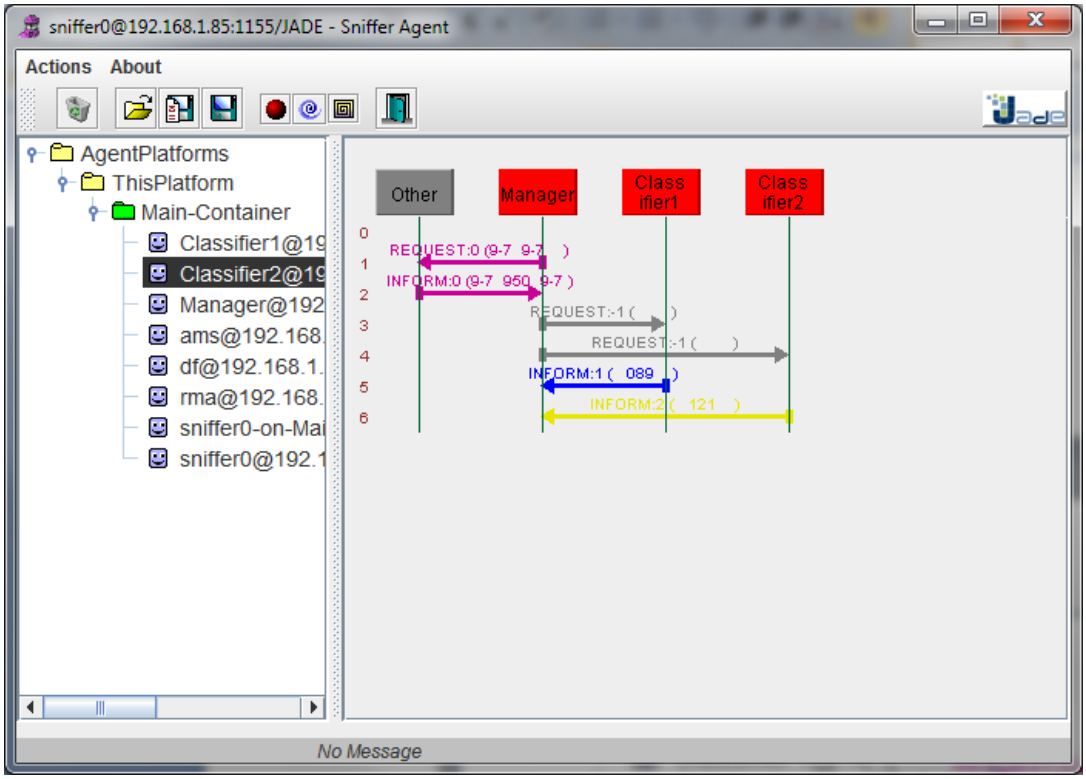
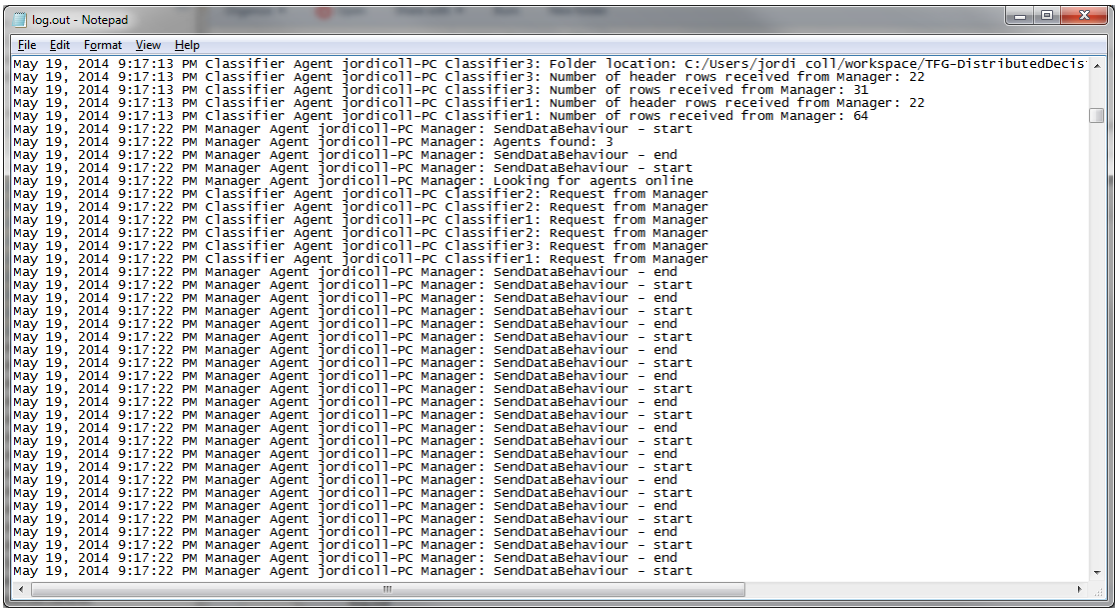


Figura 41 Mostra comunicació entre agents Jade

8.3 Debugging i Logging

Tal com s'ha especificat en l'apartat de la configuració, disposem de dues propietats en els agents que permeten mostrar més informació per pantalla i que es pot desar directament a un fitxer .out mitjançant log4j.



```
May 19, 2014 9:17:13 PM Classifier Agent jordicoll-PC Classifier3: Folder location: C:/Users/jordi coll/workspace/TFG-DistributedDecis
May 19, 2014 9:17:13 PM Classifier Agent jordicoll-PC Classifier3: Number of header rows received from Manager: 22
May 19, 2014 9:17:13 PM Classifier Agent jordicoll-PC Classifier3: Number of rows received from Manager: 31
May 19, 2014 9:17:13 PM Classifier Agent jordicoll-PC Classifier1: Number of header rows received from Manager: 22
May 19, 2014 9:17:13 PM Classifier Agent jordicoll-PC Classifier1: Number of rows received from Manager: 64
May 19, 2014 9:17:22 PM Manager Agent jordicoll-PC Manager: SendDataBehaviour - start
May 19, 2014 9:17:22 PM Manager Agent jordicoll-PC Manager: Agents found: 3
May 19, 2014 9:17:22 PM Manager Agent jordicoll-PC Manager: SendDataBehaviour - end
May 19, 2014 9:17:22 PM Manager Agent jordicoll-PC Manager: SendDataBehaviour - start
May 19, 2014 9:17:22 PM Manager Agent jordicoll-PC Manager: Looking for agents online
May 19, 2014 9:17:22 PM Classifier Agent jordicoll-PC Classifier2: Request from Manager
May 19, 2014 9:17:22 PM Classifier Agent jordicoll-PC Classifier2: Request from Manager
May 19, 2014 9:17:22 PM Classifier Agent jordicoll-PC Classifier1: Request from Manager
May 19, 2014 9:17:22 PM Classifier Agent jordicoll-PC Classifier2: Request from Manager
May 19, 2014 9:17:22 PM Classifier Agent jordicoll-PC Classifier3: Request from Manager
May 19, 2014 9:17:22 PM Classifier Agent jordicoll-PC Classifier1: Request from Manager
May 19, 2014 9:17:22 PM Manager Agent jordicoll-PC Manager: SendDataBehaviour - end
May 19, 2014 9:17:22 PM Manager Agent jordicoll-PC Manager: SendDataBehaviour - start
May 19, 2014 9:17:22 PM Manager Agent jordicoll-PC Manager: SendDataBehaviour - end
May 19, 2014 9:17:22 PM Manager Agent jordicoll-PC Manager: SendDataBehaviour - start
May 19, 2014 9:17:22 PM Manager Agent jordicoll-PC Manager: SendDataBehaviour - end
May 19, 2014 9:17:22 PM Manager Agent jordicoll-PC Manager: SendDataBehaviour - start
May 19, 2014 9:17:22 PM Manager Agent jordicoll-PC Manager: SendDataBehaviour - end
May 19, 2014 9:17:22 PM Manager Agent jordicoll-PC Manager: SendDataBehaviour - start
May 19, 2014 9:17:22 PM Manager Agent jordicoll-PC Manager: SendDataBehaviour - end
May 19, 2014 9:17:22 PM Manager Agent jordicoll-PC Manager: SendDataBehaviour - start
May 19, 2014 9:17:22 PM Manager Agent jordicoll-PC Manager: SendDataBehaviour - end
May 19, 2014 9:17:22 PM Manager Agent jordicoll-PC Manager: SendDataBehaviour - start
May 19, 2014 9:17:22 PM Manager Agent jordicoll-PC Manager: SendDataBehaviour - end
May 19, 2014 9:17:22 PM Manager Agent jordicoll-PC Manager: SendDataBehaviour - start
May 19, 2014 9:17:22 PM Manager Agent jordicoll-PC Manager: SendDataBehaviour - end
May 19, 2014 9:17:22 PM Manager Agent jordicoll-PC Manager: SendDataBehaviour - start
May 19, 2014 9:17:22 PM Manager Agent jordicoll-PC Manager: SendDataBehaviour - end
May 19, 2014 9:17:22 PM Manager Agent jordicoll-PC Manager: SendDataBehaviour - start
May 19, 2014 9:17:22 PM Manager Agent jordicoll-PC Manager: SendDataBehaviour - end
May 19, 2014 9:17:22 PM Manager Agent jordicoll-PC Manager: SendDataBehaviour - start
May 19, 2014 9:17:22 PM Manager Agent jordicoll-PC Manager: SendDataBehaviour - end
May 19, 2014 9:17:22 PM Manager Agent jordicoll-PC Manager: SendDataBehaviour - start
May 19, 2014 9:17:22 PM Manager Agent jordicoll-PC Manager: SendDataBehaviour - end
May 19, 2014 9:17:22 PM Manager Agent jordicoll-PC Manager: SendDataBehaviour - start
```

Figura 42 Exemple fitxer de logging.

9 Conclusions

9.1 Estudi

Aquest projecte està concebut com a una plataforma de codi obert i està implementada utilitzant l'entorn de desenvolupament d'agents Jade, Java i Weka per als algoritmes classificadors. La idea bàsica és la de la generació de diferents agents classificadors disjunts on cada agent disposarà d'informació diferent sobre les dades a analitzar i per tant un punt de vista diferent. Cada agent disposa d'un algorisme de classificació diferent amb la possibilitat de *fine-tuning*.

S'intenta doncs, aconseguir un sistema amb una capacitat de raonament més elevat independentment de les dades d'entrada. Al finalitzar l'execució, es pot veure el resultat de la classificació i s'entrega un informe amb les dades provinents de cada classificador així com la decisió final presa mitjançant el mecanisme de consens.

9.1.1 Repositoris online.

Un dels avantatges del projecte és que es pot utilitzar qualsevol repositori de dades que vulguem per a l'anàlisi. El sistema està preparat per a modificar les dades en calent, és a dir, el sistema permet:

- Reentrenar els agents classificadors.
- Reenviament de les dades a classificar o afegir més dades.
- Creació d'agents al vol i executar les dues operacions anteriors.

Amb aquest model d'arquitectura, aconseguim una flexibilitat molt alta que ens permetrà una integració molt elevada amb diversos sistemes.

9.2 Reptes

En aquest apartat s'expressen algun dels reptes trobats a l'hora de realitzar la integració de tots els components utilitzats en aquest projecte i quina ha sigut la seva resolució per a corregir-los. D'aquesta manera també es transfereix el coneixement après i es mostren els resultats apresos per a una base sòlida del coneixement en cas de que algú vulgui continuar evolucionant aquest projecte.

- **Protégé i Bean Generator**:
    - Si s'utilitza la última versió disponible de Protégé, aquesta no serà compatible amb *Bean Generator* i no es podrà crear l'ontologia per Jade. Per a realitzar-ho cal anar a una versió antiga de Protégé com la versió 3.4.5 ja que aquesta disposa l'arquitectura de llibreries en el mateix format que *Bean Generator*. *Bean Generator* v3.4.1 és compatible amb aquesta última versió de Protégé i tot i fer servir la última versió de Jade (v4.3.1) les classes generades son totalment compatibles.

- **Integració Weka i Jade**:
    - Cal configurar correctament el CLASSPATH per a que el funcionament de la llibreria sigui correcte, sinó errors del tipus *classnotfoundexception* apareixeran quan un dels clients vulgui interaccionar amb el Weka.
    - La última versió del Weka v3.7.10 canvia amb el tema del DataSource i disposa d'un mòdul que permet la interacció amb diferents fitxers de dades. Al carregar aquest mòdul des de el Jade, aquest tornarà una excepció del tipus *cannot find module core.weka* etc. Per tant després d'investigar una mica, cal que la generació del weka es faci amb una versió anterior, en aquest cas la v3.6.6 que només accepta fitxers ARFF com a entrada de dades i que funciona perfectament amb Jade v4.3.1.

- **JADE repetició de missatges**:
    - S'ha comprovat un funcionament estrany a l'hora d'executar els behaviours cíclics i es retornen missatges duplicats a les cues dels agents. Això implica que quan un agent envia un objecte a una de les cues dels agents, aquest objecte està duplicat a la cua. He enviat un missatge als desenvolupadors del Jade per veure si és realment un bug o simplement algun problema a l'hora de definir els behaviours. Aquest és el missatge que s'ha enviat a jade-develop@avalor.tilab.com:

Hi All,

I'm using 2 cyclic behaviours to send messages between agents. One sends a REQUEST while the other replies back with an INFORM. The problem I'm facing here is that the second agent is replying back twice and I get 2 identical responses in the queue.

Agent 1 Agent 2
REQUEST-------->
<----------INFORM
<----------INFORM

I can always filter the response if this is a bug, but the funny thing is that if I'm trying to add more agents, for example one agent asking 3 more agents, they reply with a different number of responses. In this case one agent will reply back with 6 repeated answers, another with 4 and the last one with 2. I haven't tried adding another agent, but I guess that there is some sort of problem here.

I don't know what else to check as I'm just following simple REQUEST-INFORM examples but still I can find duplicates in the queues. I've done a workaround about it but I would be grateful if someone could give me a hint of what's going on. I can provide some code if necessary.

Regards,
Jordi

- **Configuració i codificació dels agents**:
    - Degut a la dificultat de poder depurar el codi d'una manera senzilla des de l'interfície de desenvolupament, tot es fa més llarg i cada error costa molt més de trobar. D'aquí el fet d'haver d'usar proves unitàries per assegurar que qualsevol tasca de *refactoring* no trenques el codi.

9.3 Resultats

L'objectiu d'aquest projecte és el d'una avaluació empírica sobre la generació d'una arquitectura amb agents distribuits mitjançant jade i la implementació del weka per a la generació d'agents classificadors independents per obtenir una infraestructura de decisió on l'estimació de la precisió és molt més elevada.

Per a l'avaluació del sistema, s'ha executat un test usant 5 agents amb diversos classificadors i amb un conjunt d'entrenament diferent per així donar diversos punts de vista o diferent decisió.

En un primer test, amb el 100% de les dades d'entrenament i amb un classificador simple, el sistema obtenia 6 discrepàncies:

6.7,3.0,5.0,1.7,Iris-versicolor	Classifier1 (weka.classifiers.lazy.IBk) 1ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms **disagreed
6.1,3.0,4.9,1.8,Iris-virginica	Classifier1 (weka.classifiers.lazy.IBk) 1ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 1ms **disagreed
6.0,2.7,5.1,1.6,Iris-versicolor	Classifier1 (weka.classifiers.lazy.IBk) 0ms | -> Final Decision using OWA (Iris-virginica) value:2.00 in 0ms **disagreed
6.3,2.8,5.1,1.5,Iris-virginica	Classifier1 (weka.classifiers.lazy.IBk) 1ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms **disagreed
6.1,2.6,5.6,1.4,Iris-virginica	Classifier1 (weka.classifiers.lazy.IBk) 1ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 1ms **disagreed
6.0,3.0,4.8,1.8,Iris-virginica	Classifier1 (weka.classifiers.lazy.IBk) 1ms | -> Final Decision using OWA (Iris-versicolor) value:1.00 in 0ms **disagreed

Aquestes discrepàncies ens indiquen que la decisió presa per l'agent és diferent de la classificació original feta, per tant d'acord amb les dades que l'agent té, ha definit que aquesta classificació és incorrecte i n'ha donat una de nova.

Ara bé, al fer servir un model diferent, amb 5 agents classificadors amb un conjunt diferent de dades, obtenim un sistema de discriminació diferent.

Com es pot veure en els resultats de la sortida de l'execució del test 2, només s'han trobat 4 discrepàncies:

```
6.7,3.0,5.0,1.7,Iris-versicolor
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-virginica(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-virginica(92.00%) 1ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-versicolor(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-virginica(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(76.00%) 0ms
                Final Decision using OWA (Iris-virginica) value:1.62 in 0ms **disagreed
6.0,2.7,5.1,1.6,Iris-versicolor
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-virginica(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-virginica(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-versicolor(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-virginica(76.00%) 0ms
                Final Decision using OWA (Iris-virginica) value:1.59 in 1ms **disagreed
6.3,2.8,5.1,1.5,Iris-virginica
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-virginica(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-versicolor(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-versicolor(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(76.00%) 1ms
                Final Decision using OWA (Iris-versicolor) value:1.21 in 1ms **disagreed
6.1,2.6,5.6,1.4,Iris-virginica
        Classifier1 (weka.classifiers.trees.J48) decision:Iris-virginica(84.00%) 0ms
        Classifier2 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(92.00%) 0ms
        Classifier3 (weka.classifiers.functions.MultilayerPerceptron) decision:Iris-virginica(73.33%) 0ms
        Classifier4 (weka.classifiers.trees.J48) decision:Iris-versicolor(69.33%) 0ms
        Classifier5 (weka.classifiers.lazy.IBk) decision:Iris-versicolor(76.00%) 0ms
                Final Decision using OWA (Iris-versicolor) value:1.40 in 1ms **disagreed
```

Per tant, tenim 2 elements de l'anterior resultat que si que son corroborats, obtenint així un sistema amb una precisió més elevada.

En l'àrea de l'intel·ligència artificial, es coneix que l'ús de tècniques de classificació basades en reconeixement de patrons i mètodes d'agrupació defineixen una discriminació major i aconsegueixen així millors respostes sobre les classificacions.

L'objectiu d'aquest projecte és el de poder generar aquestes eines que poden ajudar a aprofundir en l'anàlisi d'aquestes dades, millorant la classificació de les dades a analitzar i sobretot usant la potència de l'intel·ligència artificial en quant a construir agents que poder ser entrenats a voluntat per aconseguir unes classificacions molt més valuoses i amb molta més precisió.

El sistema pot ser desplegat en xarxa i cada agent disposa d'un classificador que queda a l'espera de les ordres del manager. Cada agent conté un sistema de reentrenament sobre les dades a tractar i això el fa molt més intel·ligent a l'hora de prendre decisions i mostrar una discriminació molt més acurada.

En el nostre cas, el sistema fa una discriminació molt més bona sobre el tipus de flor segons les dades d'aquestes i indica que la classificació inicial no és correcta basant-se

en la informació que hem aconseguit entrenant els agents i amb la idea que s'han format sobre aquests conjunts. Per tant creen un sistema d'inferència on generen prediccions locals i s'envien a l'agent manager per a la decisió final.

Aquest projecte està enfocat per tractar qualsevol tipus de dades i en qualsevol àmbit, d'aquí l'especial interès que pot aconseguir en diferents sectors i sobretot la viabilitat d'obtenir un sistema capaç de generar decisions basant-se en entrenament.

Tal i com es comentava en el projecte HealthAgents, aquests mencionen que "*un sistema basat en agents distribuïts intel·ligents poden produir sistemes de programari innovadors que poden ajudar a solucionar grans problemes*"[9].

9.4 Millores futures

Aquí es recopilen una sèrie de millores que es podrien realitzar sobre la plataforma:

- Configuració de la plataforma per a un sistema distribuït mitjançant paràmetres de configuració remota.
- Gestió dels errors de comunicacions. Si un agent falla, el sistema queda a l'espera d'una resposta. Si l'agent no respon, el sistema es queda penjat.

10 Glossari

JADE = Java Agent DEvelopment Framework.

POC = Proof of concept.

WOWA = Weighted Ordered Weighted Average.

ARFF = Attribute-Relation File Format.

CSV = Comma-Separated values.

FIPA = Foundation for Intelligent Physical Agents

IMTP = Internal Message Transport Protocol

AMS = Agent Management System

DF = Directory Facilitator

RMI = Remote method invocation

ACC= Agent Communication Channel

ACL = Agent Communication Language

11 Eines d'implementació

- **Eclipse** modeling Kepler https://www.eclipse.org. Programming IDE, *Kepler Service Release 2*.
- **Eclipse Pluggins**:
  - **Papyrus 0.10.2.v201402191554**: http://www.eclipse.org/papyrus/ Graphical editor tool for UML2.
  - **Graphical Editing Framework** v**3.9.1.201308190730** http://www.eclipse.org/gef/
  - **ObjectAid v1.1.5** http://www.objectaid.com/
  - **SWT v4.3** http://www.eclipse.org/swt/
- **Weka** v3.7.10 (31/07/2013). http://sourceforge.net/projects/weka/files/weka-3-7-windows-x64/3.7.10/.
- **Jade** v4.3.1 (05/12/2013). http://jade.tilab.com/maven/com/tilab/jade/jade/4.3.1/.
- **Protégé v3.4.5**. http://protege.stanford.edu/
- **Bean Generator for Protege v3.2.1**. http://protege.cim3.net/cgi-bin/wiki.pl?OntologyBeanGenerator
- **Log4j v1.2.17**: http://www.apache.org/dyn/closer.cgi/logging/log4j/1.2.17/log4j-1.2.17.zip

12 Bibliografia

[1] González-Vélez H, Mier M, Julià-Sapé M, N. Arvanitis T, García-Gómez J, Robles M, H. Lewis P, Dsmahapatra S, Dupplaw D, Peet A, Arús C, Celda B, Van Huffel S, Luuch-Ariet M, (1997) HealthAgents: Distributed multi-agent brain tumor diagnosis. In: Springer Science+Business Media, LLC 2007.

[2] Multi-Agent Systems [Consulta en línia] http://jasss.soc.surrey.ac.uk/4/2/reviews/rouchier.html [Data de consulta: 07/03/2014]

[3] Merigó J, A unified model between the weighted average and the induced OWA operator. Expert systems with Applications 2011.

[4] Nettleton D, Torra V, (2001) A comparison of Active Set Method and Genetic Algorithm Approaches for Learning Weighting Vectors in Some Aggregation Operators. International Journal of Intelligent Systems, John Wiley & Sons, Inc 2001.

5 Jade Agent Development Framework [Consulta en línia] http://jade.tilab.com/ [Data de consulta: 19/05/2014]

6 Bellifemine F, Caire G, Grenwood D, (2004) Developing multi-Agent Systems with Jade. John Wiley & Sons Ltf, The Atrium, Southern Gate, Chichester 2004.

7 Weka 3: Data Mining Software in Java [Consulta en línia] http://www.cs.waikato.ac.nz/ml/weka/ [Data de consulta: 01/04/2014]

[8] Merigó J, A unified model between the weighted average and the induced OWA operator. Expert systems with Applications 2011.

[9] González-Vélez H, Mier M, Julià-Sapé M, N. Arvanitis T, García-Gómez J, Robles M, H. Lewis P, Dsmahapatra S, Dupplaw D, Peet A, Arús C, Celda B, Van Huffel S, Luuch-Ariet M, (1997) HealthAgents: Distributed multi-agent brain tumor diagnosis. In: Springer Science+Business Media, LLC 2007