Sesión 5 - Integración KNIME y Weka

Sistemas de ayuda a la toma de decisiones

Alejandro Piedrafita Barrantes
Alejandro Ruiz Sumelzo
José Ignacio Hernández Gracia
Alberto Martínez Rodríguez
Alex Daniel Costa
Víctor Hernández Fernández
Jorge Fernández Muñoz

Universidad Zaragoza

Enero de 2020

Índice

Página

1. Descripción del problema

El problema que se plantea es realizar un flujo de trabajo que produzca las mismas tareas que en el ejercicio 3 de la sesión 3 (entrenamiento de modelo *Decision Tree J48*, *Multilayer Perceptron* y *Naive Bayes*.

Tomando como conjunto de datos de entrenamiento los ficheros *winde.data*, *iris.data* y *adult.data*) se realizarán varias ejecuciones con proporciones de entrenamiento distintas y se presentarán los resultados de manera gráfica.

2. Arquitectura del sistema

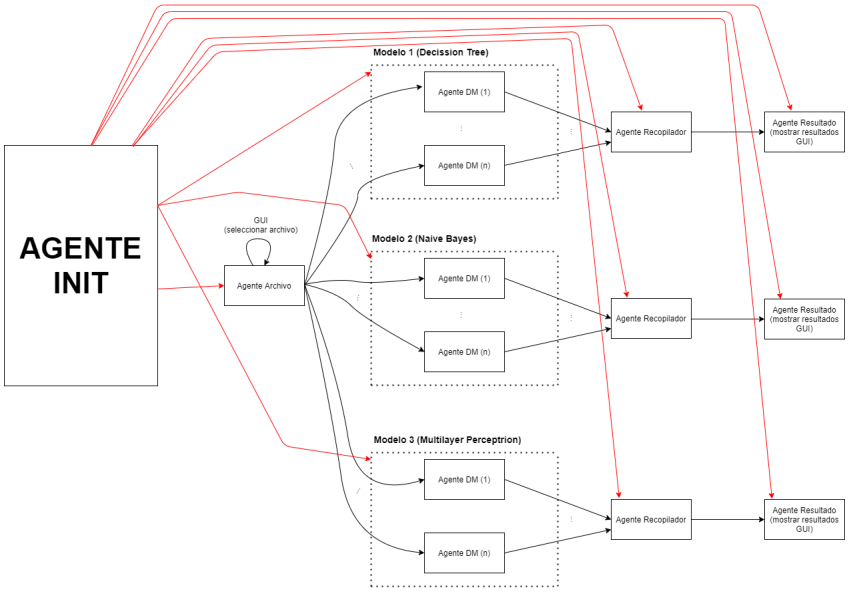

Figura 1: Propuesta de arquitectura del sistema

2.1. Propuesta de solución

La propuesta consiste en introducir dentro del sistema de *Agentes* utilizados, un *Agente* que realice las labores de inicialización del trabajo, haciendo uso de las funciones que permiten a un agente crear dinámicamente nuevos agentes, con el fin de que, dado que los agentes que realizan las distintas tareas tienen comportamientos de tipo *OneShotBehaviour*, sea capaz de generar nuevos agentes que vayan completando las distintas tareas que son necesarias para la posterior evaluación de todos los resultados.

Además, se introducirán en el sistema *Agentes* encargados de la recopilación

de los resultados de los distintos procesos clasificadores con el fin de procesarlos y generar los datos resultantes que serán posteriormente enviados al *AgenteResultado*, encargado de mostrar los resultados finales.

2.2. Sociedad de agentes

2.2.1. *AgenteInit*

Agente que se encarga de crear dinámicamente todos los agentes necesarios para el flujo de trabajo, especificará el número de ejecuciones para cada modelo de entrenamiento y el porcentaje de datos que utilizará para entrenar el modelo.

2.2.2. *AgenteArchivo*

Agente que representa la *GUI* inicial, permite seleccionar el archivo de datos a tomar como entrenamiento y se lo comunica a los agentes *AgenteDM*.

2.2.3. *AgenteDM*

Agente que recibe el fichero proporcionado por la *GUI AgenteArchivo*, implementa *Weka* y realiza una iteración para clasificar mediante el modelo de decisión pertinente (*Decision Tree J48*, *Multilayer Perceptron* o *Naive Bayes*) para un porcentaje de entrenamiento (30, 50 u 80) indicado por la invocación del *AgenteInit*. El resultado se lo envía al *AgenteRecopilador*.

2.2.4. *AgentesRecopilador*

Agentes encargados de la recopilación y agrupación de los datos referidos a cada uno de los modelos que se pretenden analizar.

2.2.5. *AgenteResultado*

Agente que recibe el resultado proporcionado por los diferentes *AgenteRecopilador* y lo hace visible en el área de texto indicada por el frame (*JframAgenteResultado*).

2.3. Protocolo de Comunicación

El *AgenteInit* crea dinámicamente un *AgenteArchivo*, *número de iteraciones* x 3 (debido a utilizar tres porcentajes de partición de entrenamiento distinto) agentes *AgenteDM* para cada modelo de entrenamiento, tres *AgenteRecopilador* (uno para cada modelo) y tres *AgenteResultado*.

Este agente obtendrá el controlador del contenedor en el cual ha sido creado y lo utilizará para crear los distintos agentes. Los agentes serán creados de atrás hacia adelante (primero el resultado y último el archivo) para asegurarse de que sus respectivas comunicaciones se realizan de forma correcta y no se pierden mensajes debido a que los agentes no han sido creados aún.

A su vez, en cada creación de los *AgenteDM*, se especificará mediante un argumento inicial el porcentaje de partición que le corresponde utilizar.
AgenteDM en su creación se queda bloqueado, será *AgenteArchivo* quien le proporcione el fichero de datos que debe utilizar para hacer la clasificación. Tiene comportamiento *OneShot*, por lo que una vez finalizada la clasificación, enviará una única vez los resultados al *AgenteRecopilador* correspondiente a su modelo.

El *AgenteRecopilador* tiene un comportamiento cíclico. Se quedará bloqueado inicialmente a la espera de recibir mensajes de los agentes correspondientes a un mismo modelo, para cada ejecución que reciba clasificará los resultados por el porcentaje de partición y hará un promedio total. Estos tres resultados (uno por cada porcentaje de partición de los datos de entrenamiento) los enviará al *AgenteResultado* correspondiente para que los muestre.

AgenteResultado recibirá los resultados pertinentes al modelo que representa y los mostrará mediante una interfaz gráfica.

El protocolo de comunicación utilizado para los mensajes intercambiados entre los agentes en esta arquitectura será el protocolo *FIPA Request Interaction Protocol*. Dichos mensajes se enviarán de izquierda a derecha atendiendo a la disposición de los agentes en la arquitectura (archivo hace una petición a *DM*, *DM* a *Recopilador* y *Recopilador* a *Resultado*). El agente que inicia la comunicación enviará el *request* inicial al participante y esperará su confirmación. El participante, tras recibir la petición devuelve un mensaje de confirmación (*agree*) para notificar al emisor que ha recibido el mensaje correctamente.

Tras ello, realizará la tarea concreta que le haya pedido el emisor y finalmente notificará al emisor de que ha acabado dicha tarea mediante un *inform-done*. Al realizar las peticiones de izquierda a derecha, los emisores no necesitan un resultado por lo que no es necesario utilizar *inform-result*.

Además, en caso de que alguno de los mensajes no llegue y no se reciba o bien confirmación o bien respuesta, se establecerá un *timeout*; el cual, en caso de expirar, provocará el reintento de dicha acción (hasta que esta se acabe llevando a cabo finalmente).

2.4. Dinámica de funcionamiento

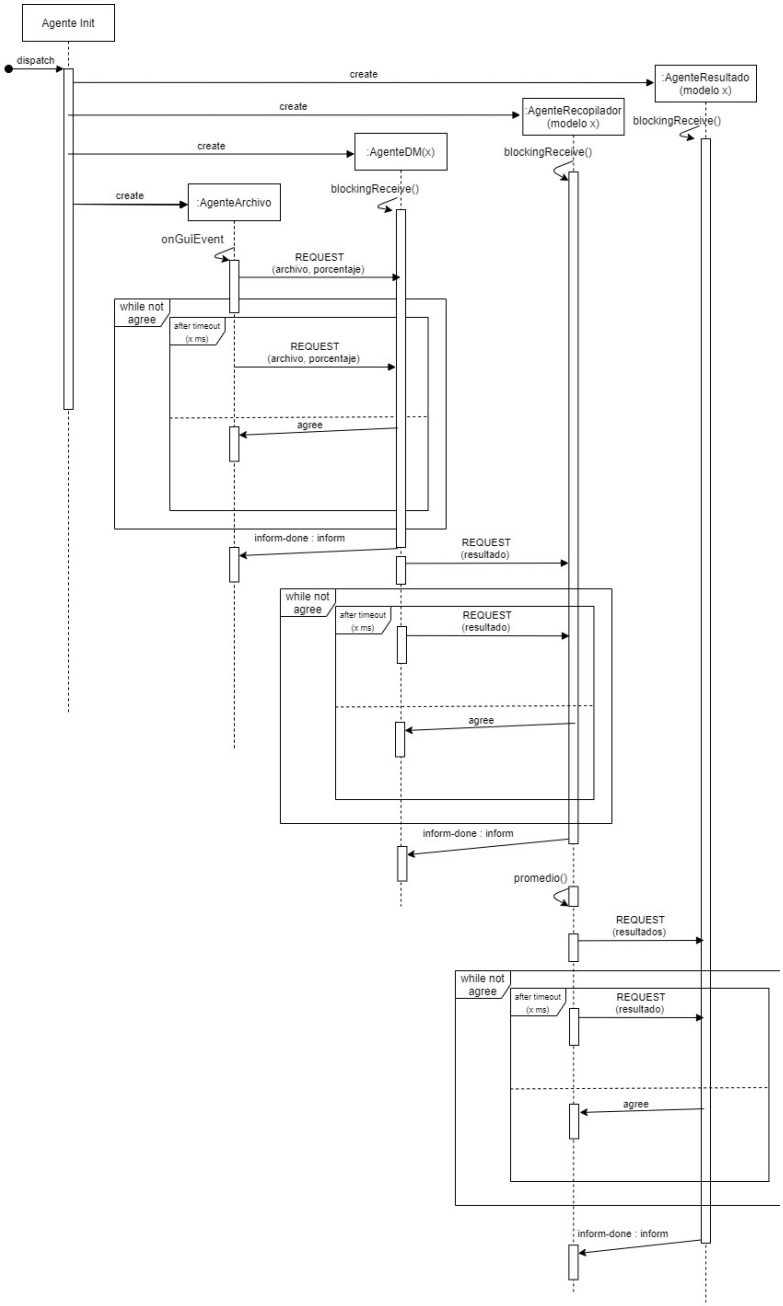

Figura 2: Diagrama de secuencias de la solución propuesta

3. Experimentos

3.1. Descripción

Los experimentos abordados en este proyecto atienden a dos criterios: iteraciones por archivo y archivos existentes.

En primer lugar, para comprobar el correcto funcionamiento de los diferentes agentes y la comunicación entre los mismos, se utilizó una única iteración y se modificó el *timeout* en diversas ejecuciones para comprobar tanto el correcto envío y recepción de mensajes como los *timeouts* en caso de no cumplir con los tiempos establecidos.

Tras haber determinado que el escenario creado y los distintos agentes eran correctos, se aumentaron las iteraciones (dos, tres y finalmente cinco iteraciones por modelo) para así finalmente poder comparar con los resultados de lo implementado en *Knime* en la tercera práctica de la asignatura (en la cual se realizaban cinco iteraciones). Estas pruebas se han realizado para cada uno de los tres archivos disponibles en el material de este proyecto, lo que ha supuesto la ejecución en el entorno de *Knime* de las mismas pruebas con estos ficheros, con el fin de obtener valores con los cuales comparar.

3.2. Comparación de resultados con la práctica 3

Para la realización de las pruebas de nuestro sistema, se han utilizado los 3 ficheros de extensión .arff facilitados, de los cuales se han realizado el mismo número de iteraciones que se realizaron en su día en la practica 3. Debido a que estos ficheros utilizados no son los mismos que en su día fueron utilizados en la practica 3, se realizaron, tomando como base el sistema diseñado en la practica 3 de Knime, las mismas pruebas con estos ficheros en Knime, dando como resultado de ambos los siguientes resultados comparativos:

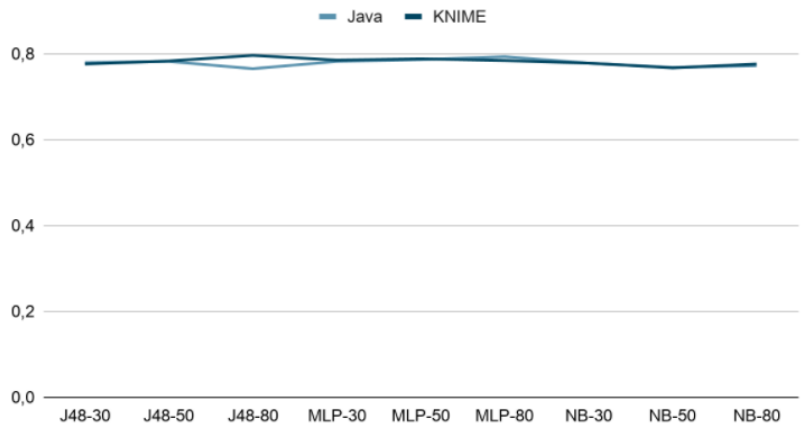

Figura 3: Gráfica comparativa de tiempos entre *KNIME* y el sistema, con los datos de *archivo.arff*

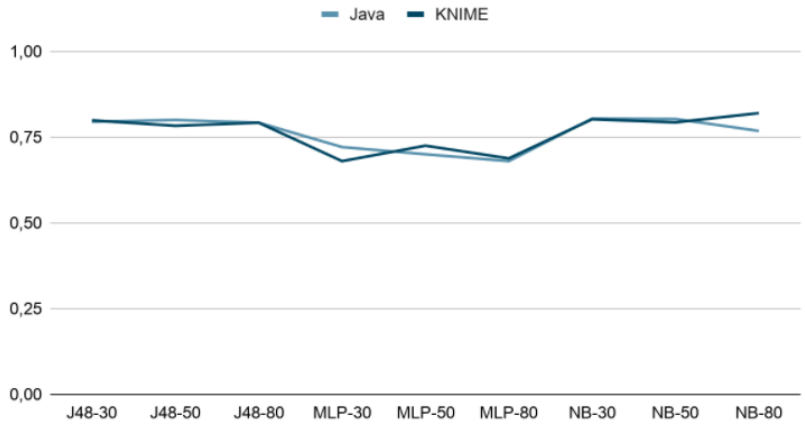

Figura 4: Gráfica comparativa de tiempos entre *KNIME* y el sistema, con los datos de *arqueologia.arff*

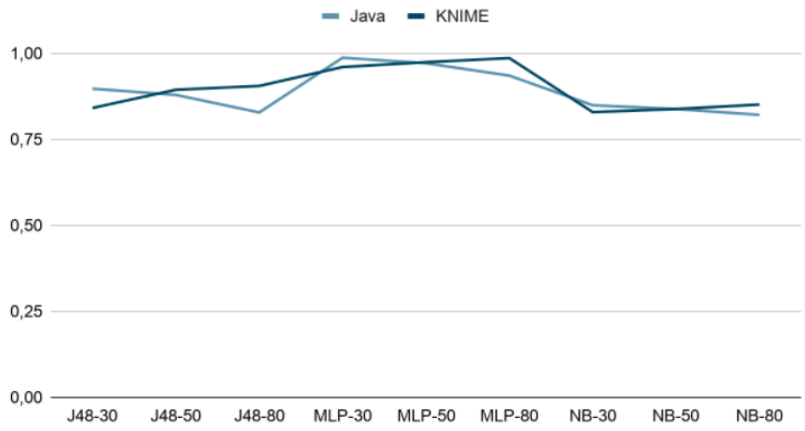

Figura 5: Gráfica comparativa de tiempos entre KNIME y el sistema, con los datos de *vehiculo.arff*

En general se puede observar como los datos resultantes son muy similares en ambos casos, exceptuando alguna situación en la cual existe una diferencia ligeramente mayor, pero esto puede ser debido a múltiples razones, la principal de ellas es que se realizan únicamente 5 iteraciones, por lo cual una iteración con un resultado anómalo afecta de manera más alta que en el caso de haber utilizado un mayor número de iteraciones.

3.3. Pruebas de rendimiento (local)

Para las pruebas de rendimiento se ha utilizado una máquina con 4 núcleos y 12 GB de RAM, en la cual los resultados obtenidos en relación al tiempo empleado en ejecutar cada uno de los modelos ha sido incremental, tomando como ejemplo la ejecución del fichero *vehiculos.arff*, en la cual la ejecución de los modelos Naive Bayes ha llevado entre 1600 y 2000 ms, la ejecución de los modelos J48 entre 2500 y 2700 ms y por ultimo, la ejecución de los modelos del Multilayer, entre 90000 y 159000 ms, por lo cual se puede ver una enorme diferencia entre este último en comparación con los demás en cuanto al tiempo de ejecución. Estas diferencias en relación al tiempo de ejecución de los modelos Multilayer aparecían, en mayor o menor medida, en todas las ejecuciones del sistema.

4. Conclusiones

4.1. Grupo

La organización del trabajo se dividido en tres grupos (dentro del mismo) que atacaban los pilares del sistema. El primero se encargó de la inicialización y paso del archivo, otro de el *AgenteDM* que procesa ese archivo y lo evalúa con

los parámetros que se le indique, y por último el agente encargado de recopilar los resultados de todos los *AgenteDM* de manera que presenta la media a través de interfaz gráfica. Para la coordinación en el paso de mensajes fue necesario acordar entre todos la sintaxis de mensaje, así como el protocolo que seguirían.

4.2. Individual

4.2.1. José Ignacio Hernández Gracia

A nivel individual debo decir que me parece un escenario de problema asequible para familiarizarse con la comunicación entre agentes. Bien es cierto que nos retrasó el desconocimiento de la librería de jade para el paso de mensaje y saber el qué estaba ya implementado o qué lógica debíamos establecer nosotros. Estoy contento con lo aprendido y considero que será de utilidad para afrontar la segunda parte del trabajo, ya que en esta debemos pensar desde cero y con total libertad la comunicación entre agentes. En cuanto a la coordinación del equipo opino que se trabajó bien y sacó partido de las horas que se nos ofreció durante las clases para desarrollar gran parte del trabajo, no obstante fue necesario sacar más reuniones para solventar problemas que se presentaron al hacer pruebas del sistema.

4.2.2. Alberto Martínez Rodríguez

Entiendo que la finalidad de este primer proyecto es aplicar y afianzar los conocimientos dados en el tema de sistemas multiagente (sobre todo a la hora de la comunicación entre agentes) a la vez que se mejora la capacidad de trabajo en grupo, objetivo que ha sido logrado. Por otro lado no le he visto mucha utilidad a la parte que tiene que ver con los modelos de weka.

En cuanto al trabajo en grupo, creo que nos hemos coordinado correctamente a pesar de ser un grupo tan grande. Sin embargo, aun podemos mejorar en algunos aspectos como la organización y división del trabajo.

4.2.3. Alex Daniel Costa

Viniendo de otra universidad, (Erasmus - Unimore Italia) donde no se practica tanto el trabajo en grupo, puedo decir que me coordiné bien con el resto de miembros del grupo. Los sistemas *multi-agente* me parecen un tema muy interesante y este proyecto me ayudó mucho entender bien qué son, como se comportan y cómo se comunican entre ellos. También me pareció muy útil usar el código para hacer lo que hacíamos antes con *KNIME*.

4.2.4. Alejandro Piedrafita Barrantes

Este proyecto ha servido para comprender mejor el funcionamiento de los agentes JADE y los protocolos de comunicación en los mismos. También se ha aprendido a integrar *Weka* en este proyecto, útil para realizar experimentos sobre conjuntos de datos con diferentes modelos y ver así cuáles ofrecen mejores

resultados y se adaptan mejor a cada experimento.

Este proyecto me ha parecido interesante y sobre todo útil, pues se ha podido ver el propósito final de lo aprendido y una aplicación que podría ser un caso real sobre el que trabajar en el futuro laboral. Además ha sido necesario coordinarse en un grupo amplio, trabajando en grupos más reducidos y acordando la forma de comunicarse entre los distintos agentes abordados por cada grupo y juntar los resultados finalmente para conseguir un proyecto operativo, también próximo a un posible caso real.

4.2.5. Jorge Fernández Muñoz

Todas las conclusiones que saco de este proyecto son positivas, desde la orientación del mismo como una introducción al trabajo con los agentes de Jade y la comunicación entre los mismos, pasando por el número de personas que componemos el grupo del mismo, dado que es un número superior al habitual y requiere de una mayor coordinación, todo lo cual nos ayudará de aquí a futuro, y además se trata de un proyecto que incluye mucha comunicación entre procesos, lo que también ha repercutido en una parte muy importante de coordinación para la comunicación entre los diversos procesos del proyecto.

4.2.6. Víctor Hernández Fernández

Acostumbrados a grupos pequeños y a prácticas de corto desarrollo, este proyecto nos ha permitido experimentar y desarrollar coordinación y comunicación, junto a la optimización de cómo dividir los problemas a resolver y, posteriormente, la unión de trabajo y las pruebas para verificar su correcto funcionamiento.

Además, hemos podido enfrentarnos a un problema interesante con el fin de poder aprender cómo funcionan los agentes, y como se pueden implementar y configurar para resolver el ejercicio planteado.

4.2.7. Alejandro Ruiz Sumelzo

En primer lugar, la organización y disposición de todo el grupo ha sido notable, ya que en todas las reuniones del proyecto se encontraban varias personas que iban anotando y explicando los pasos y puntos completados (o que faltaban por completar).
Esta coordinación ha repercutido en un menor tiempo completado por parte de todos los integrantes a la hora de realizar cada uno de los apartados, ya que todos sabían qué realizar en cada momento.

Trabajar con *JADE* ha permitido complementar los conocimientos sobre la plataforma, además de implementar y desarrollar un trabajo más amplio respecto a código y coordinación.
Que hubiera varias partes, y luego unir y que funcionara fue algo tedioso, a la

vez que reconfortante. Además, comparar este trabajo con otras herramientas (en este caso *KNIME*) sirve para valorar y sopesar diferentes variantes a la hora de trabajar en un entorno laboral.

Anexo I. Reuniones.

Fecha de reunión	Asistentes	Decisiones tomadas
25/11/2020 8:00 - 10:00 h	José Ignacio Hernández Alberto Martínez Alex Daniel Costa Alejandro Piedrafita Jorge Fernández	Se ha formado el grupo de trabajo. Se ha descargado el código y se ha abierto el proyecto para posteriormente estudiar sus archivos. Tras identificar los distintos agentes se ha ejecutado y se han realizado diversas pruebas para comprender el funcionamiento.
27/11/2020 9:30 - 11:30 h	José Ignacio Hernández Víctor Hernández Alex Daniel Costa Alejandro Piedrafita Alejandro Ruiz Jorge Fernández	Se ha identificado el problema a abordar y se ha realizado una configuración inicial de la sociedad de agentes. Se ha diseñado un esquema que representa la arquitectura final, así como el comportamiento de cada uno de los agentes.
30/11/2020 10:00 - 11:00 h	José Ignacio Hernández Alberto Martínez Alex Daniel Costa Alejandro Piedrafita Jorge Fernández Alejandro Ruiz Víctor Hernández	Se mantiene una retrospectiva con el profesor que indica la necesidad de introducir un diagrama de comunicación y la descripción verbal de qué protocolo se utiliza para la comunicación entre agentes.
02/12/2020 11:00 - 12:00 h	José Ignacio Hernández Alberto Martínez Alex Daniel Costa Alejandro Piedrafita Jorge Fernández Alejandro Ruiz Víctor Hernández	A partir de las modificaciones a realizar acordadas en la tutoría del lunes (con el profesor de la asignatura) se crea el diagrama de comunicación que representa el intercambio de mensajes entre los distintos agentes y se amplía la explicación incluyendo el protocolo seguido y el tipo de mensajes intercambiados.
04/12/2020 9:30 - 12:00 h	José Ignacio Hernández Alberto Martínez Alex Daniel Costa Alejandro Piedrafita Jorge Fernández Alejandro Ruiz Víctor Hernández	Se finaliza la construcción de la propuesta de informe para la sociedad multiagente. Tras la entrega final de la sesión 5, se comienza a configurar el proyecto (código).
09/12/2020 10:00 - 11:00 h	José Ignacio Hernández Alberto Martínez Alex Daniel Costa Alejandro Piedrafita Jorge Fernández Alejandro Ruiz Víctor Hernández	Se realiza la migración de la memoria a *Overleaf*. Se asignan las tareas concretas de código para la realización de la primera parte (además de Weka).
21/12/2020 16:00 - 18:00 h	José Ignacio Hernández Alex Daniel Costa Alejandro Piedrafita	Se ha trabajado en el código del Agente DM, programándolo según lo acordado en las reuniones grupales y el diagrama de funcionamiento, a falta únicamente de una correcta gestión de los timeouts.
18/12/2020 14:00 - 16:30 h	Alex Daniel Costa Alberto Martínez	Se ha desarrollado el AgenteRecopilador.

Fecha de reunión	Asistentes	Decisiones tomadas
28/12/2020 16:00 - 17:00 h	José Ignacio Hernández Alejandro Ruiz Sumelzo Alejandro Piedrafita	Se ha concluido el código del Agente DM.
29/12/2020 10:00 - 13:00 h	José Ignacio Hernández Alex Daniel Costa Alejandro Piedrafita Jorge Fernández Alejandro Ruiz Víctor Hernández	Se acaba el código de todo el proyecto. Se prueba cada uno de los agentes y su uso correcto.

Anexo II. Tablas de tiempos.

Modelo y proporción entrenamiento	Resultados de 5 iteraciones	Media
Decision Tree J48 30%	0.791 | 0,776 | 0.783 | 0.772 | 0.775	0.779
Decision Tree J48 50%	0.780 | 0.781 | 0.782 | 0.781 | 0.786	**0.782**
Decision Tree J48 80%	0.739 | 0.780 | 0.770 | 0.768 | 0.768	0.765
Multilayer Perceptron 30%	0.787 | 0.787 | 0.779 | 0.780 | 0.776	0.782
Multilayer Perceptron 50%	0.784 | 0.773 | 0.793 | 0.788 | 0.794	0.786
Multilayer Perceptron 80%	0.802 | 0.807 | 0.809 | 0.775 | 0.773	**0.793**
Naive Bayes 30%	0.768 | 0.778 | 0.781 | 0.787 | 0.775	**0.778**
Naive Bayes 50%	0.763 | 0.767 | 0.781 | 0.750 | 0.777	0.768
Naive Bayes 80%	0.773 | 0.800 | 0.768 | 0.741 | 0.777	0.772

Figura 6: Tabla de tiempos de *archivo.arff*

Modelo y proporción entrenamiento	Resultados de 5 iteraciones	Media
Decision Tree J48 30%	0.784 | 0.775 | 0.789 | 0.759 | 0.775	0.776
Decision Tree J48 50%	0.792 | 0.779 | 0.774 | 0.789 | 0.779	0.783
Decision Tree J48 80%	0.816 | 0.798 | 0.775 | 0.816 | 0.775	**0.796**
Multilayer Perceptron 30%	0.785 | 0.792 | 0.781 | 0.785 | 0.784	0.785
Multilayer Perceptron 50%	0.795 | 0.790 | 0.781 | 0.783 | 0.790	**0.788**
Multilayer Perceptron 80%	0.759 | 0.802 | 0.780 | 0.793 | 0.786	0.784
Naive Bayes 30%	0.778 | 0.775 | 0.781 | 0.778 | 0.780	**0.778**
Naive Bayes 50%	0.742 | 0.791 | 0.779 | 0.782 | 0.742	0.767
Naive Bayes 80%	0.791 | 0.791 | 0.759 | 0.778 | 0.759	0.776

Figura 7: Tabla de tiempos de *archivo.arff* con *KNIME*

Modelo y proporción entrenamiento	Resultados de 5 iteraciones	Media
Decision Tree J48 30%	0.793 | 0,770 | 0.816 | 0.804 | 0.793	0.795
Decision Tree J48 50%	0.774 | 0.822 | 0.806 | 0.806 | 0.790	**0.800**
Decision Tree J48 80%	0.840 | 0.720 | 0.720 | 0.760 | 0.920	0.792
Multilayer Perceptron 30%	0.735 | 0.712 | 0.701 | 0.758 | 0.701	**0.721**
Multilayer Perceptron 50%	0.580 | 0.774 | 0.741 | 0.645 | 0.758	0.700
Multilayer Perceptron 80%	0.560 | 0.600 | 0.800 | 0.720 | 0.720	0.680
Naive Bayes 30%	0.758 | 0.827 | 0.827 | 0.839 | 0.770	**0.804**
Naive Bayes 50%	0.774 | 0.822 | 0.806 | 0.822 | 0.790	0.803
Naive Bayes 80%	0.840 | 0.920 | 0.760 | 0.560 | 0.760	0.768

Figura 8: Tabla de tiempos de *arqueologia.arff*

Modelo y proporción entrenamiento	Resultados de 5 iteraciones	Media
Decision Tree J48 30%	0.804 | 0.781 | 0.804 | 0.804 | 0.804	**0.799**
Decision Tree J48 50%	0.790 | 0.806 | 0.806 | 0.806 | 0.709	0.783
Decision Tree J48 80%	0.800 | 0.760 | 0.800 | 0.800 | 0.800	0.792
Multilayer Perceptron 30%	0.609 | 0.689 | 0.712 | 0.735 | 0.655	0.680
Multilayer Perceptron 50%	0.774 | 0.774 | 0.596 | 0.709 | 0.774	**0.725**
Multilayer Perceptron 80%	0.800 | 0.680 | 0.720 | 0.560 | 0.680	0.688
Naive Bayes 30%	0.781 | 0.816 | 0.804 | 0.804 | 0.804	0.802
Naive Bayes 50%	0.822 | 0.806 | 0.806 | 0.725 | 0.806	0.793
Naive Bayes 80%	0.800 | 0.840 | 0.800 | 0.820 | 0.840	**0.820**

Figura 9: Tabla de tiempos de *arqueologia.arff* con *KNIME*

Modelo y proporción entrenamiento	Resultados de 5 iteraciones	Media
Decision Tree J48 30%	0.894 | 0,904 | 0.906 | 0.887 | 0.896	**0.897**
Decision Tree J48 50%	0.853 | 0.886 | 0.879 | 0.876 | 0.901	0.879
Decision Tree J48 80%	0.832 | 0.832 | 0.829 | 0.826 | 0.823	0.828
Multilayer Perceptron 30%	0.989 | 0.986 | 0.988 | 0.985 | 0.989	**0.987**
Multilayer Perceptron 50%	0.967 | 0.973 | 0.984 | 0.965 | 0.969	0.971
Multilayer Perceptron 80%	0.945 | 0.930 | 0.907 | 0.962 | 0.933	0.935
Naive Bayes 30%	0.835 | 0.856 | 0.852 | 0.865 | 0.841	**0.849**
Naive Bayes 50%	0.841 | 0.836 | 0.848 | 0.829 | 0.837	0.838
Naive Bayes 80%	0.819 | 0.823 | 0.826 | 0.820 | 0.826	0.821

Figura 10: Tabla de tiempos de *vehiculo.arff*

Modelo y proporción entrenamiento	Resultados de 5 iteraciones	Media
Decision Tree J48 30%	0.826 | 0.845 | 0.859 | 0.853 | 0.825	0.841
Decision Tree J48 50%	0.898 | 0.892 | 0.893 | 0.895 | 0.894	0.894
Decision Tree J48 80%	0.930 | 0.907 | 0.898 | 0.893 | 0.901	**0.905**
Multilayer Perceptron 30%	0.951 | 0.960 | 0.961 | 0.957 | 0.973	0.960
Multilayer Perceptron 50%	0.971 | 0.971 | 0.975 | 0.982 | 0.974	0.974
Multilayer Perceptron 80%	0.994 | 0.976 | 0.988 | 0.991 | 0.983	**0.986**
Naive Bayes 30%	0.842 | 0.842 | 0.820 | 0.821 | 0.824	0.829
Naive Bayes 50%	0.846 | 0.837 | 0.828 | 0.835 | 0.846	0.838
Naive Bayes 80%	0.832 | 0.861 | 0.861 | 0.855 | 0.846	**0.851**

Figura 11: Tabla de tiempos de *vehiculo.arff* con *KNIME*