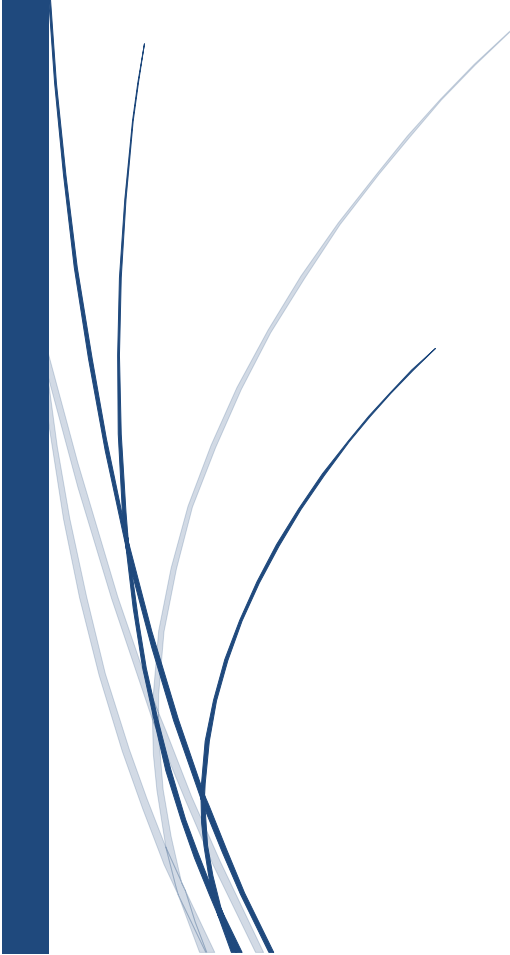




18-1-2016

Programación de un sistema de agentes

Aprendizaje de modelos de entrenamiento para conjuntos de datos reales.



Guillermo Pérez – 610382
Sandra Campos – 629928
Andrea Aleyxendri – 626549
Alejandro Bean – 625788

Contenido

Introducción	2
Modelo	2
Decisiones de diseño	3
Prototipo.....	3
Implementación en Knime	3
Implementación en JADE	5
Evaluación.....	6
Descripción de los experimentos	6
Resultados obtenidos con Knime	7
Resultados obtenidos con JADE	10
Gestión del proyecto	19
Organización del trabajo	19
Herramientas utilizadas.....	19
Diagrama GANTT	19
Conclusiones	20
Valoración colectiva	20
Valoraciones individuales	20
Bibliografía y enlaces de interés	21

Introducción

Este proyecto está dedicado a la programación de un sistema de agentes con la herramienta JADE para mejorar el entrenamiento de seis conjuntos de datos (cuatro obligatorios y dos opcionales) en comparación con las herramientas que hemos utilizado durante el curso como Knime y Weka. No obstante, tanto Knime como Weka siguen siendo utilizados en este trabajo para generar los workflows necesarios.

Los objetivos del trabajo son los siguientes:

- Desarrollar los workflows en Knime, utilizando nodos de Weka, para posteriormente entrenarlos y probarlos.
- Programar el sistema JADE para que tome los datos de los workflows generados en la tarea anterior y contrastar con los resultados obtenidos.
- Conclusiones generadas de ambos modelos.

Modelo

Los tres modelos obligatorios utilizados han sido los siguientes:

- **J48:** es una implementación open-source en java para el algoritmo de árboles de decisión C4.5. Se hizo muy popular en 2008 cuando fue nombrado #1 en el Top 10 de algoritmos para minería de datos.[2]

Este algoritmo construye árboles de decisión desde un grupo de datos de entrenamiento de la misma forma en que lo hace ID3, usando el concepto de entropía de información.

- **Naive Bayes:** es un algoritmo probabilístico de clasificación fundamentado en el teorema de Bayes. Su principal característica es que asume que la presencia o ausencia de una característica particular no está relacionada con la presencia o ausencia de cualquier otra característica, dada una clase variable.
- **ID3:** El algoritmo ID3 es utilizado dentro del ámbito de la inteligencia artificial. Su uso se centra en la búsqueda de hipótesis o reglas dado un conjunto de ejemplos basándose en un árbol de decisión. Dicho conjunto deberá tener una serie de tuplas de valores (llamados atributos) en el que uno de ellos será el objetivo, es decir, el atributo a clasificar. Teniendo en cuenta la condición de que este atributo tiene que ser binario, el algoritmo intentará obtener la hipótesis que clasifique los nuevos registros de datos y sacar otro dato de tipo binario.

Por otro lado, los opcionales elegidos han sido:

- **ZeroR:** Es uno de los algoritmo de clasificación más simples. Este algoritmo solo sirve para predecir la clase que es más común, la mayoritaria. Aunque no tiene poder de predicción, es útil para ver la tendencia base.
- **MLP:** Es una red neuronal artificial (RNA) formada por múltiples capas.

Las capas pueden clasificarse en tres tipos:

- Capa de entrada: Constituida por aquellas neuronas que introducen los patrones de entrada en la red. En estas neuronas no se produce procesamiento.
- Capas ocultas: Formada por aquellas neuronas cuyas entradas provienen de capas anteriores y cuyas salidas pasan a neuronas de capas posteriores.
- Capa de salida: Neuronas cuyos valores de salida se corresponden con las salidas de toda la red.

Su uso suele estar destinado a resolver problemas de asociación de patrones, segmentación de imágenes, compresión de datos, etc.

- **K-star[3]:** Es un clasificador que se diferencia de los demás por usar una función de distancia basada en la entropía.

Decisiones de diseño

- Se decidieron agrupar los 6 ficheros de datos en 2 metanodos, uno para los obligatorios y otro para los opcionales escogidos.
- Se empleó un nodo Normalizer para normalizar todos los atributos numéricos y también para no tener que conectar el fichero elegido con los seis metanodos de algoritmos cada vez que se decida realizar un cambio. De este modo basta con cambiar la conexión entrante al Normalizer, ya que las de salida permanecen sin cambios.
- Dentro del metanodo de cada algoritmo se decidió replicar los 5 nodos necesarios para realizar la predicción (clasificador, predictor, column filter, java y group) para así poder realizar al mismo tiempo una iteración con 3 repartos de datos distintos. En prácticas anteriores se realizó con una sola línea de trabajo y después de ejecutarla era necesario tomar nota de los resultados, cambiar el reparto en Partitioning, reejecutar y comparar los resultados.

Prototipo

Implementación en Knime

El flujo de trabajo elaborado en Knime presenta la siguiente distribución:

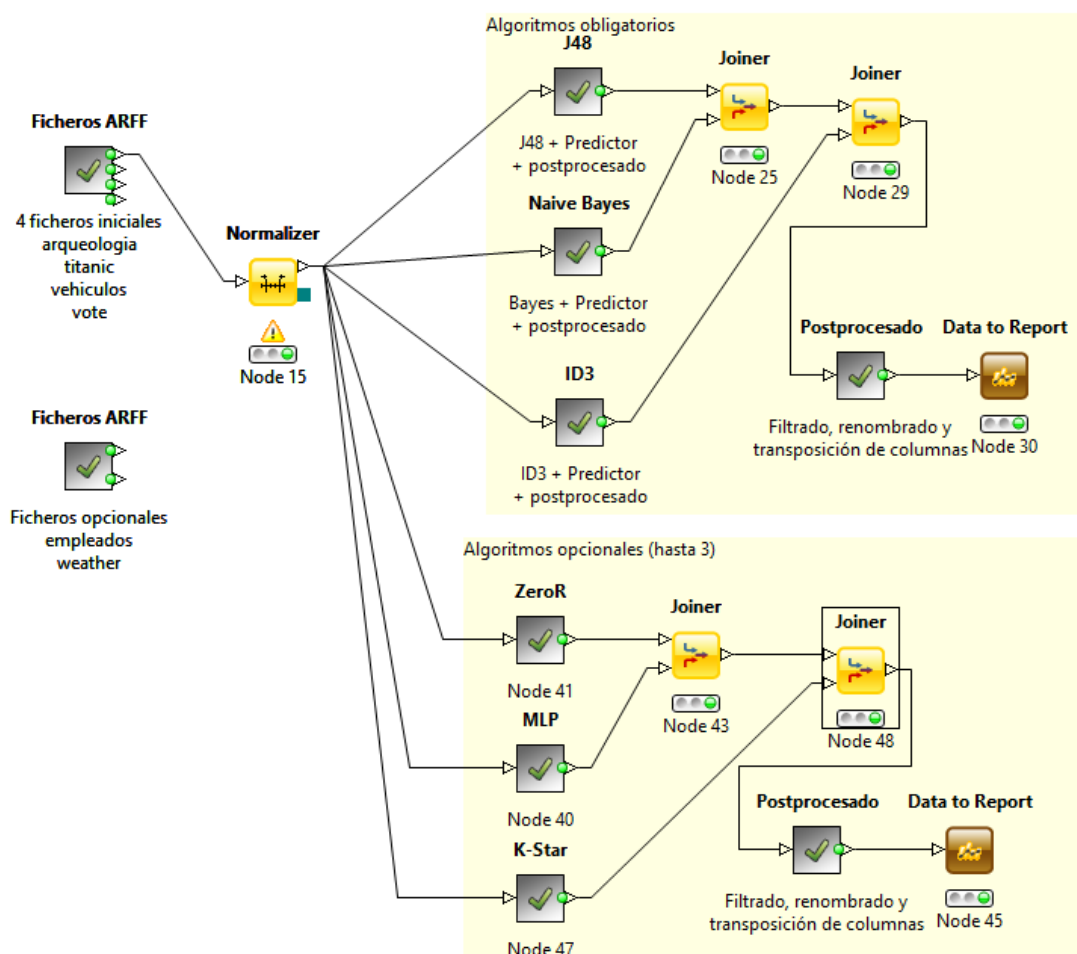


Figura 1: Distribución de nodos en el entorno de trabajo Knime

Los distintos algoritmos se agruparon en metanodos, dentro de los cuales se emplearon en tres ramas en paralelo un nodo del algoritmo (J48, naive bayes, id3, zeroR, MLP y K-estrella), un predictor de Weka, un filtro de columnas para ignorar todas menos las dos útiles de cara al Data Report, un Java Snippet y un nodo Group By para crear columna nueva y realizar el conteo. Con esta distribución de nodos se pueden realizar tres ejecuciones en paralelo con distintas particiones de datos: 20-80, 50-50 y 80-20.

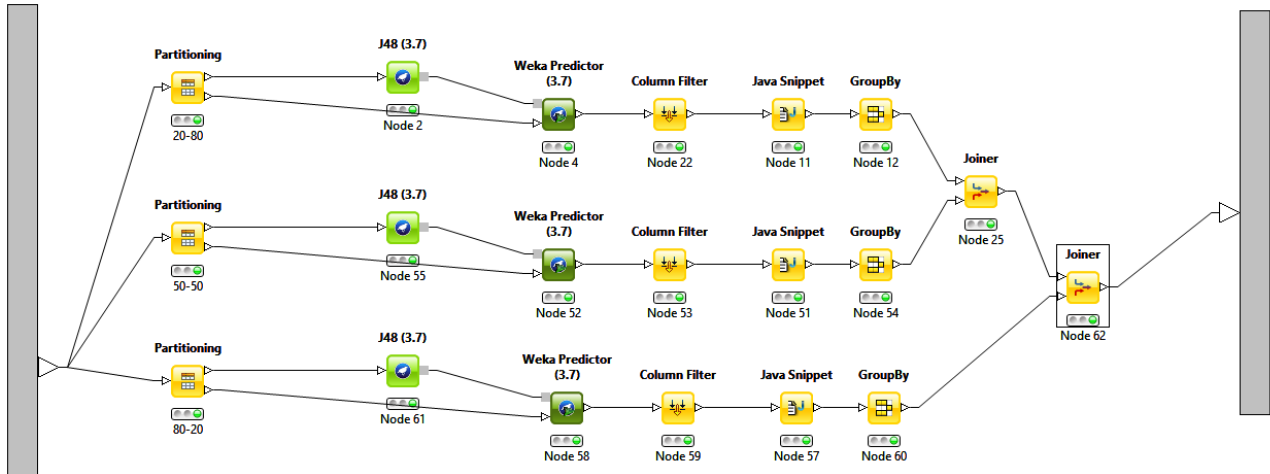


Figura 2: distribución de nodos en el metanodo del algoritmo J48

Posteriormente se realizan dos Joins para unir en una misma tabla los datos de los tres distintos repartos de datos de entrenamiento (figura 2, nodos 25 y 62), y en el exterior del metanodo de cada algoritmo se vuelven a realizar otros dos joins para unificar los datos obtenidos de cada algoritmo (figura 1, nodos 25-29 y 43-48).

Esta tabla resultante se procesa en un nodo RowID para que la clave de cada fila sea True o False (aciertos y fallos de cada método). Se renombran las columnas para acortar los nombres (en los ejes de las gráficas sólo caben nombres cortos), se transpone la tabla para poder operar la tasa de aciertos/fallos en cada fila (algoritmo y reparto de datos). Se empleó un java snippet para poner a 0 los casos en los que hubiera 0 aciertos o 0 fallos (undefined da problemas al procesar) y un column filter para que sólo quede la columna "hit/miss ratio" que queremos ver en el informe.

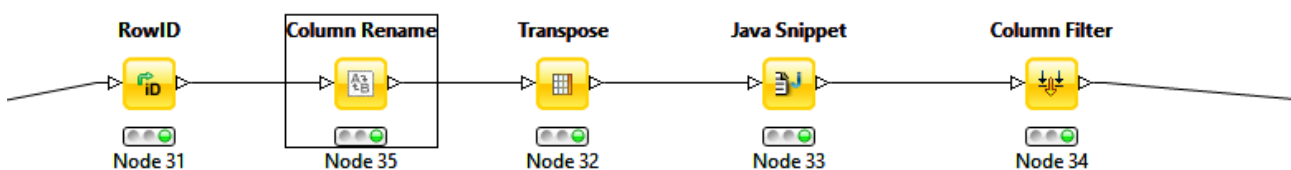


Figura 3: metanodo de postprocesado de datos

Se dividieron los resultados en dos gráficas: una para los algoritmos obligatorios y otra para los optativos.

En el caso del algoritmo ID3 se tuvo que utilizar el nodo Missing Value previamente a los distintos Partitioners para eliminar aquellas filas en las que hubiera al menos un valor nulo, puesto que este algoritmo necesita que todas las columnas tengan un valor. De las 434 filas del fichero original, sólo 123 estaban completas.

Implementación en JADE

Para la implementación del proyecto en JADE se ha utilizado como modelo el código de ejemplo proporcionado.

El proyecto se divide en 5 clases java:

- **AgenteArchivo.java:** Esta clase representa un agente que se encarga de capturar la información introducida por el usuario y enviárselo por mensaje a otro agente para que realice las correspondientes operaciones.

El usuario podrá introducir dos datos al sistema: en primer lugar la ruta con el fichero de datos y en segundo lugar se ha incorporado la posibilidad de escoger el porcentaje de datos que se puede utilizar para el entrenamiento.

El envío de información entre agentes se realiza añadiendo un tipo de dato String al contenido del mensaje, por lo que para implementar esta nueva opción se envía un mensaje compuesto, que consta de:

Porcentaje + "*" + Contenido fichero**

El carácter *** se utiliza únicamente como separador.

Se escogió """" porque eran unos caracteres que no afectaban al contenido del fichero real, por ejemplo el carácter "-" que fue la primera decisión de diseño daba problemas con alguno de los datasets que se utilizaban en el proyecto.

- **AgenteDM.java:** Este agente es la base del proyecto, en él se analizan todos los modelos con JADE y se obtienen los resultados que se enviarán a otro agente para que los muestre por pantalla.

En primer lugar se encarga de recibir el mensaje del agente anterior, y crear las instancias de los datos. Posteriormente, se le aplican dos filtros a los datos: eliminar campos nulos (importante para el algoritmo Id3 donde no se permiten valores nulos) y convertir datos numéricos en nominal. También hemos visto una buena opción mezclar los datos entre ellos para posteriormente hacer una partición para el entrenamiento de los modelos y los datos restantes para la validación.

A continuación probaremos los seis modelos escogidos: J48, MLP, Naive Bayes, K-Star, ZeroR e Id3.

Crearemos un método para cada modelo. Este método realizara las operaciones necesarias para construir un clasificador de los datos, evaluar el modelo mediante la clase *Evaluation* y devolver un entero con el número de instancias correctamente clasificadas, con el método `correct()` de la clase *Evaluation*.

Cada método se ejecutará en diez ocasiones para obtener más fiabilidad en los resultados obtenidos, por lo que tras estas operaciones obtendremos la media de las instancias correctamente clasificadas de cada modelo.

Por último el agente enviará el resultado al agente que se encarga de mostrar los resultados. Como se ha explicado anteriormente los agentes se comunican mediante mensajes donde el contenido es una cadena de caracteres, por lo que el mensaje a enviar tendrá un formato similar al anterior, con un espacio en blanco como separador.

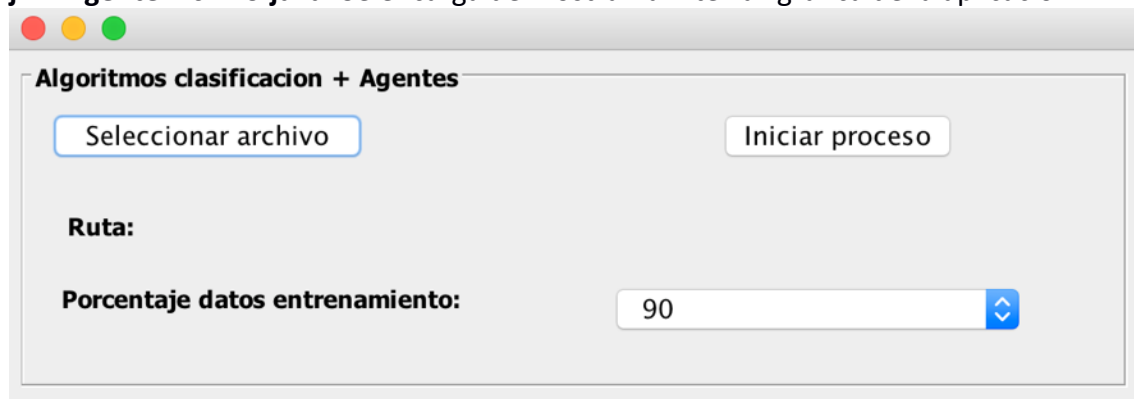
Modelo + " " + Número de Aciertos

- **AgenteResultados.java** : Este agente es el encargado de recibir los resultados del AgenteDM y mostrarlos por pantalla. Se ha optado por presentar los resultados a los usuarios en dos formatos distintos:

En primer lugar mediante histogramas, donde se pueden ver todos los modelos de forma gráfica, lo que facilita mucho la comparación entre modelos. Para implementar histogramas con JADE se hará uso de la biblioteca **JfreeChart**.

En segundo lugar se utilizará formato de texto plano, para ello se hará uso de la clase que se explicará a continuación.

- **jfrmAgenteResultado.java**: Esta clase representa la interfaz gráfica de resultados. Se encarga de mostrar el resultado de cada modelo obtenido en el agente anterior por pantalla: las instancias correctamente clasificadas, en formato de texto plano.
- **jfrmAgenteArchivo.java**: Se encarga de mostrar la interfaz gráfica de la aplicación:



- Se ha modificado para que además de escoger el fichero se pueda seleccionar el porcentaje de datos con los que poder entrenar los modelos, dando un gran abanico de posibilidades al usuario: 90, 80, 70, 60, 50, 40, 30, 20 y 10.

Evaluación

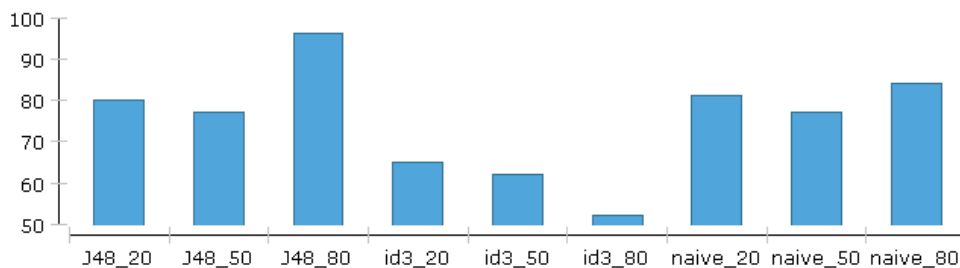
Descripción de los experimentos

Para evaluar los diferentes modelos de entrenamientos y así poder ver qué modelo es el más adecuado para cada conjunto de datos hemos realizado diferentes pruebas. Estas pruebas consisten en realizar diez ejecuciones con cada modelo y conjunto de datos. Se ha calculado la media del número de aciertos para distintas particiones de los datos para realizar el entrenamiento. A continuación se presentan los resultados de estas evaluaciones para cada conjunto de datos utilizado en forma de histograma. En cada histograma se pueden ver la media de aciertos en 10 ejecuciones para cada modelo.

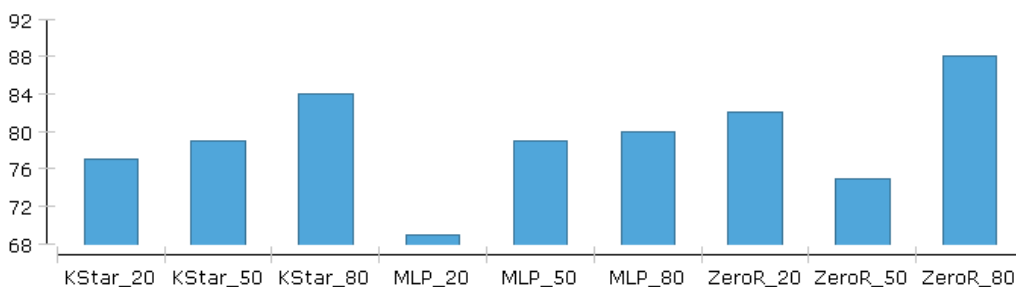
También se muestran las gráficas resultantes del flujo de trabajo de Knime. En cada gráfica se muestran 3 algoritmos y 3 repartos de datos distintos para cada uno (20% de entrenamiento, 50% y 80%).

Resultados obtenidos con Knime
Conjunto de datos: arqueologia.arff

Métodos obligatorios



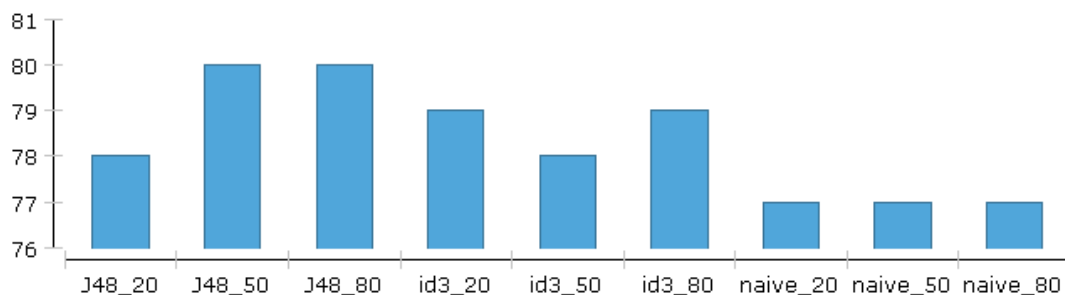
Métodos opcionales



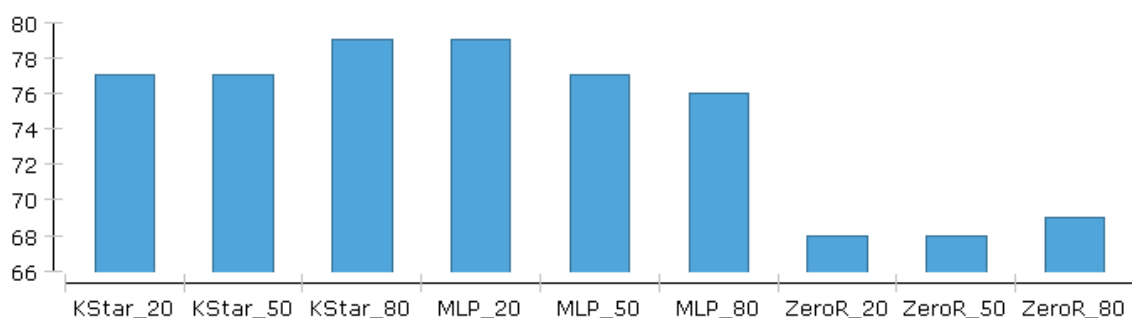
Los resultados indican una clara mejoría al emplear 80% de datos como entrenamiento en el algoritmo J48, mientras que en el ID3 pierde efectividad y en Naive Bayes los resultados son similares. MLP y K* crecen al incrementar los datos de entrenamiento. El mejor algoritmo para analizar este fichero es el árbol J48.

Conjunto de datos: Titanic.arff

Métodos obligatorios



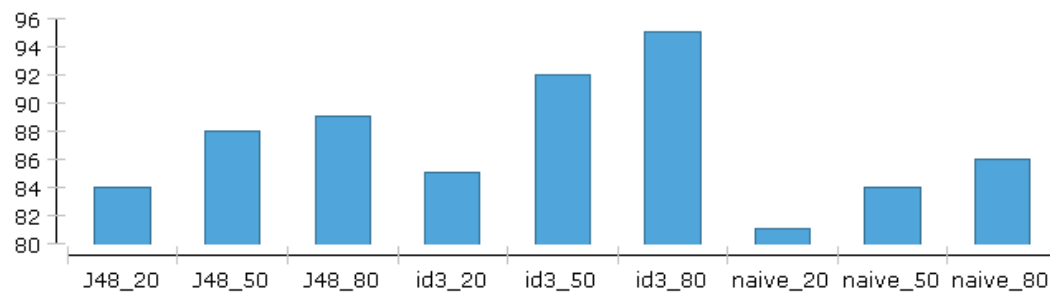
Métodos opcionales



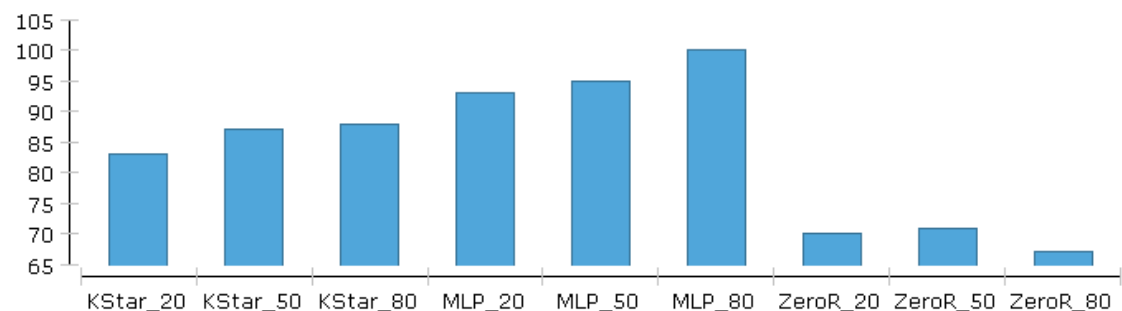
Los algoritmos J48, ID3, MLP y K* presentan los resultados más similares con una efectividad de entre el 77 y el 80% de aciertos. Los algoritmos ZeroR y Naive Bayes obtienen unos resultados pobres en comparación con los anteriores, con el 68% y 77% de aciertos respectivamente.

Conjunto de datos: vehículos.arff

Métodos obligatorios



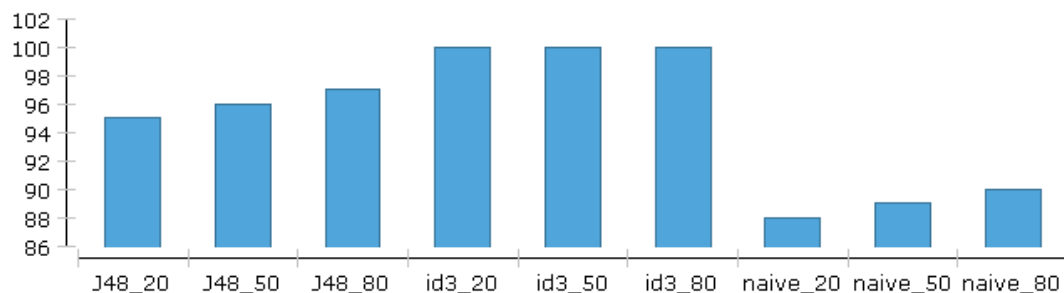
Métodos opcionales



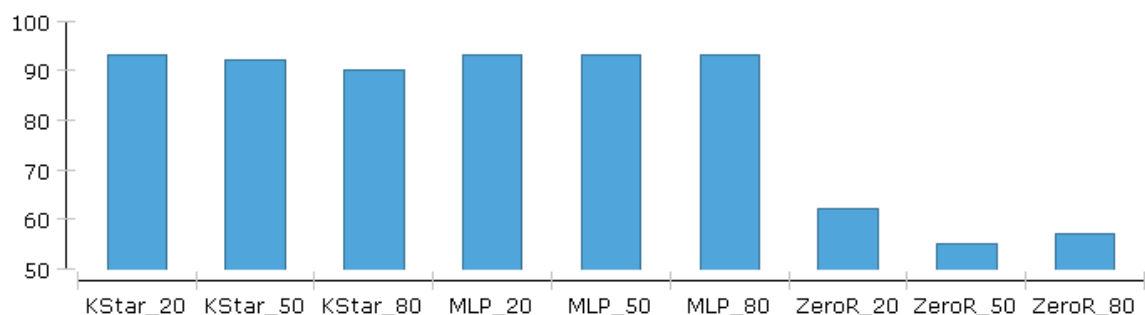
Todos los algoritmos menos el ZeroR mejoran la tasa de aciertos al incrementar la cantidad de datos de entrenamiento. Los mejores son MLP, que en el tercer caso alcanza el 100%, e ID3 que llega al 94%.

Conjunto de datos: vote.arff

Métodos obligatorios



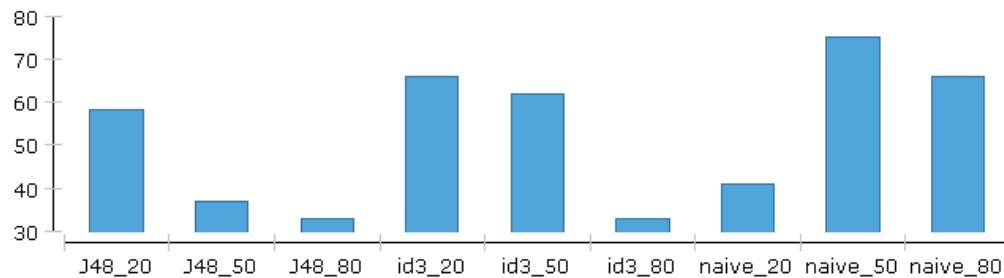
Métodos opcionales



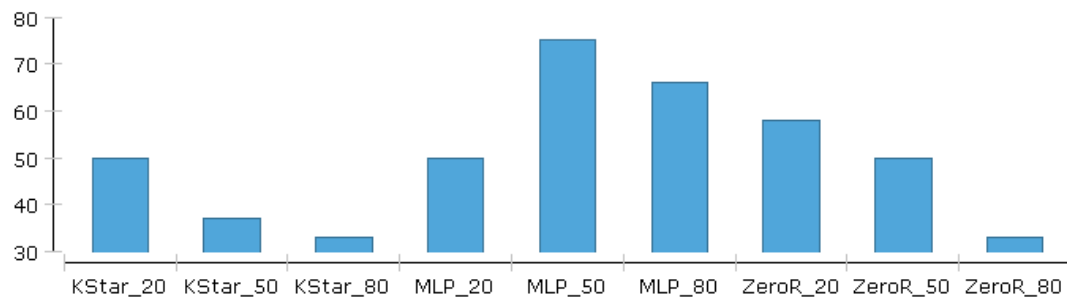
El algoritmo ID3 alcanza un 100% de acierto en los tres casos analizados. El árbol J48 también obtiene buenos resultados (95-97%) y MLP y K en torno al 90%. Naive Bayes presenta unos resultados aceptables (88-90%) pero sin duda el que peor resultados ha obtenido es ZeroR con un 55-60%.

Conjunto de datos: empleados.arff

Métodos obligatorios



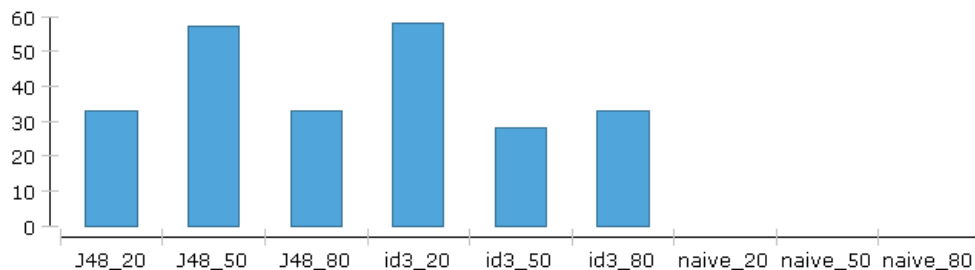
Métodos opcionales



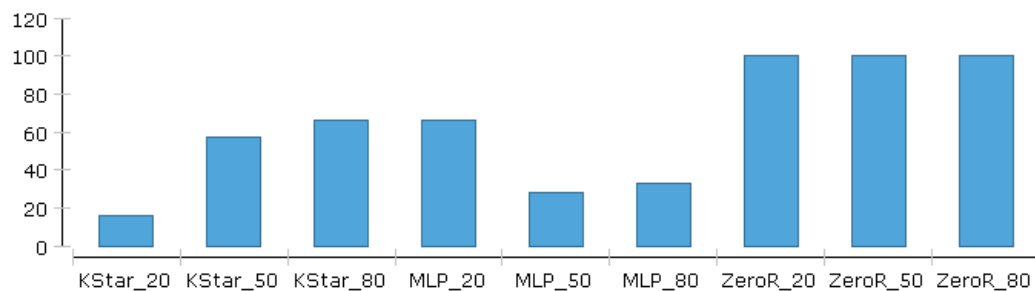
Los algoritmos J48, ID3, K* y ZeroR reducen su efectividad al aumentar la cantidad de datos de entrenamiento. Naive Bayes y MLP aumentan en el segundo caso (50% de entrenamiento) pero disminuyen ligeramente al subir hasta 80%. Los algoritmos más recomendables son estos dos últimos, MLP y Naive Bayes, al alcanzar un 75% en su mejor caso.

Conjunto de datos: weather.arff

Métodos obligatorios



Métodos opcionales



El peor algoritmo es Naive Bayes porque en los tres casos analizados no logra ningún acierto. ID3, J48, MLP y K* obtienen entre el 30% y 60%. De estos cuatro algoritmos, K* mejora al incrementar la cantidad de datos de entrenamiento y los tres restantes presentan resultados sin patrón claro. MLP e ID3 empeoran en el segundo y tercer caso, y J48 mejora en el segundo y empeora en el tercero.

Resultados obtenidos con JADE

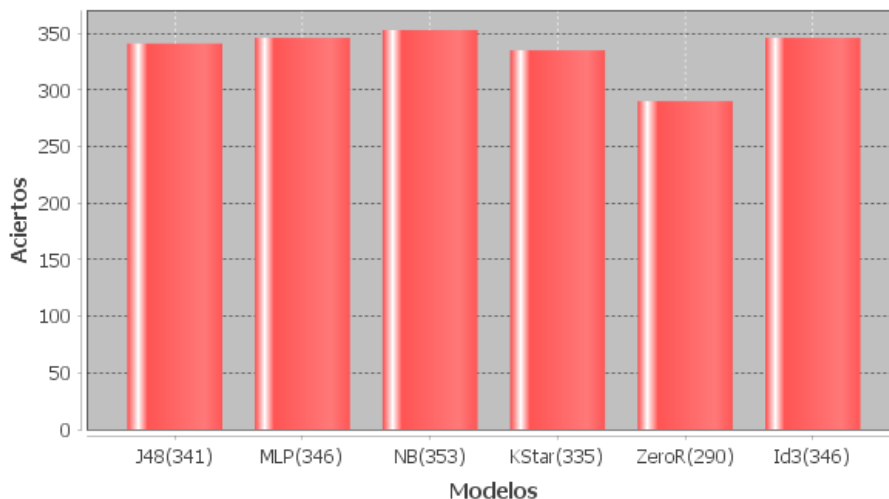
Conjunto de datos: Titanic.arff

Total de instancias: 2201.

- Partición del 80%:

El mejor modelo para este conjunto de datos y con un 80% de las instancias para el entrenamiento de los modelos de datos es Naive Bayes con una media de 353 aciertos en la clasificación del 20% restante de los datos. Naive Bayes, J48, redes neuronales (MLP) e Id3 son modelos que obtienen resultados parecidos aunque alguno destaque más que otro. El peor modelo de clasificación para este conjunto es ZeroR obteniendo una media de 290 aciertos. El modelo KStar tampoco obtiene muy buenos resultados.

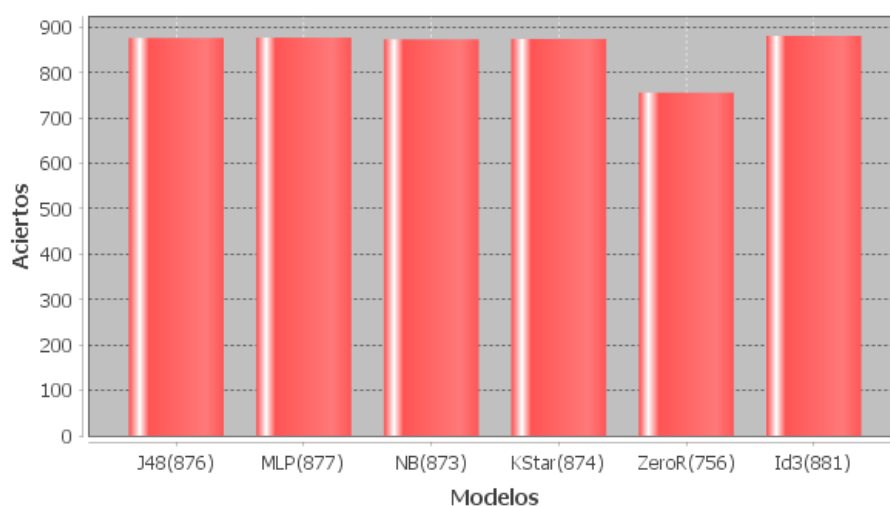
Media de aciertos en 10 iteraciones



- Partición del 50%:

Utilizando el 50% de los datos para entrenamiento y el 50% restante para realizar las pruebas o téses se consiguen unos resultados parecidos los anteriores comentados. El peor modelo sigue siendo el ZeroR con gran diferencia. Naive Bayes ya no destaca como el mejor modelo y se iguala con el resto. KStar iguala también sus resultados a los demás.

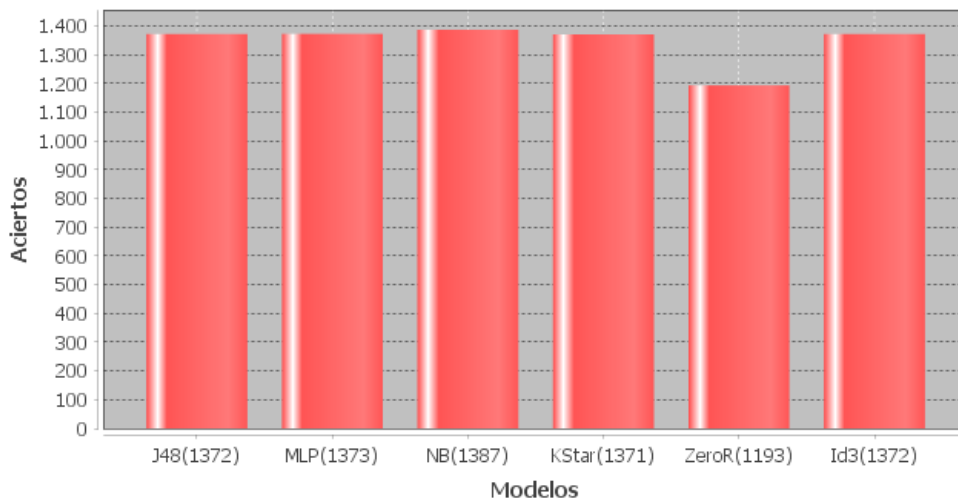
Media de aciertos en 10 iteraciones



- Partición del 20%:

Utilizando el 20% de los datos para entrenamiento se obtienen resultados equivalentes que con la partición del 50%. No existen diferencias que comentar.

Media de aciertos en 10 iteraciones



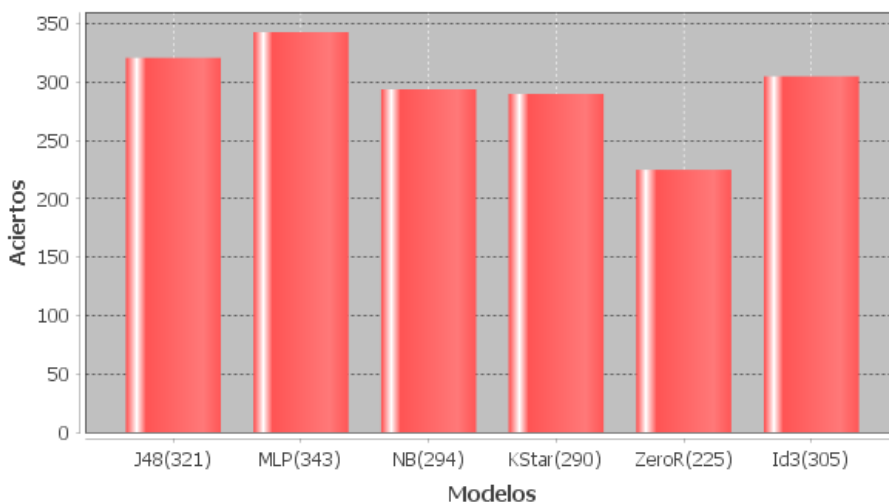
Conjunto de datos: vehículos.arff

Total de instancias: 1728.

- Partición del 80%:

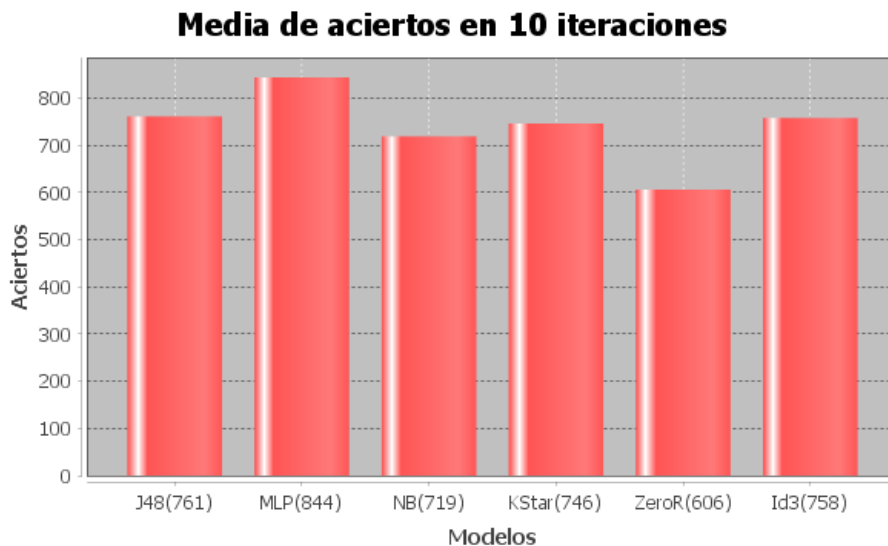
El mejor modelo es Multilayer Perceptron con una media de 343 aciertos. Le siguen los modelos de árboles de decisión (J48 e Id3). Naive Bayes está al mismo nivel que KStar y finalmente el peor modelo de entrenamiento para este conjunto es también ZeroR.

Media de aciertos en 10 iteraciones



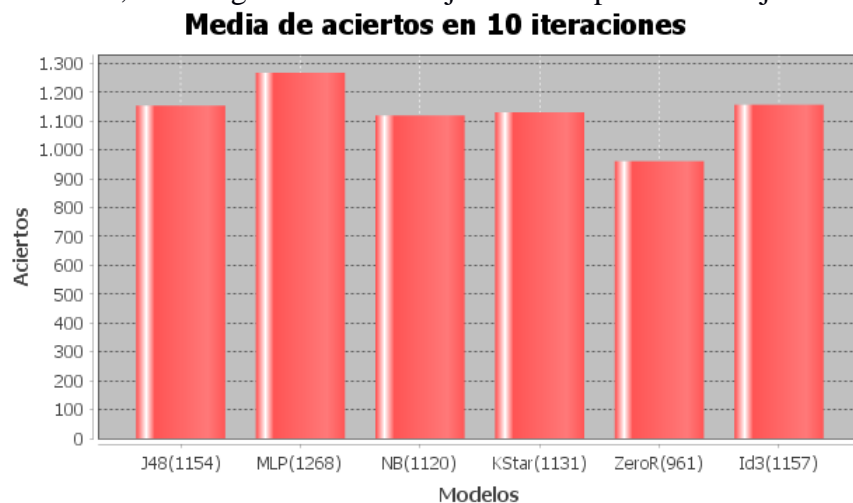
- Partición del 50%:

Los resultados son proporcionales a los resultados anteriores realizados con el 80% de los datos el entrenamiento. La única diferencia considerable es entre los modelos Naive Bayes y KStar, donde anteriormente estaban equilibrados los resultados y ahora es KStar el que parece sacarle cierta ventaja y ser mejor modelo de entrenamiento para este dataset.



- Partición del 20%:

Al igual que los resultados con particiones de datos mayores para los datos de entrenamiento, estos resultados son proporcionales a los anteriores. Naive Bayes y KStar vuelven a igualarse los resultados, MLP sigue siendo el mejor modelo para este conjunto de datos seguido por Id3 y J48.



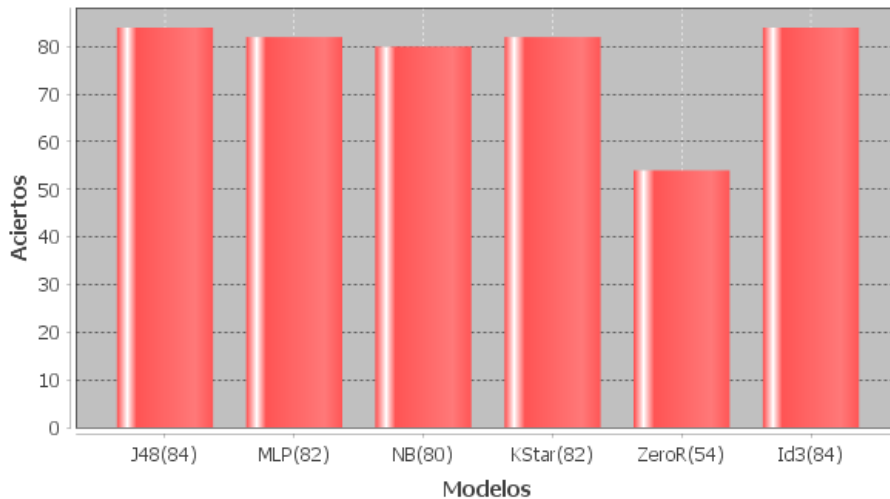
Conjunto de datos: [vote.arff](#)

Total de instancias: 435.

- Partición del 80%:

Para este conjunto de datos los mejores algoritmos que presentan mejores resultados son los de árboles de decisión (J48 e Id3). MLP, KStar y Naive Bayes también presentan resultados muy buenos. El peor algoritmo es ZeroR donde presenta resultados muy malos. Al igual que en los conjuntos de datos anteriores ZeroR sigue siendo el peor modelo de entrenamiento.

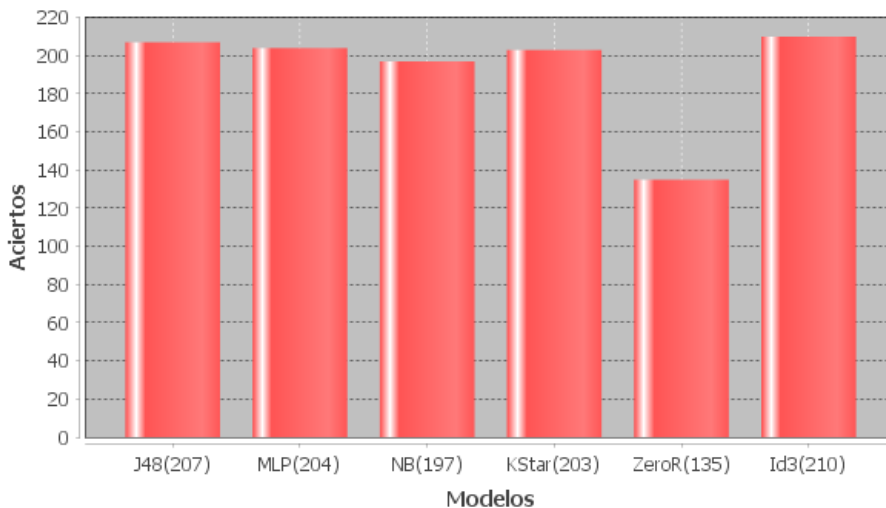
Media de aciertos en 10 iteraciones



- Partición del 50%:

Con la misma cantidad de datos para entrenamiento y para validación se presentan resultados proporcionales. ZeroR es el peor modelo con bastante diferencia y el resto están bastante equilibrados. Por poca diferencia, los mejores modelos siguen siendo Id3 y J48.

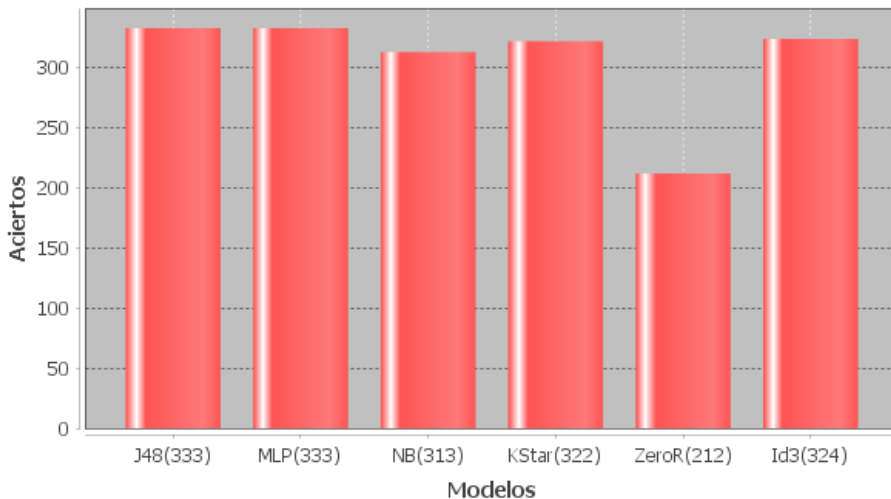
Media de aciertos en 10 iteraciones



- Partición del 20%:

En éste último caso, MLP supera a Id3 en los resultados igualándose al J48. Id3 presenta peores resultados con esta proporción. Naive Bayes está casi a la altura de KStar y finalmente ZeroR presenta resultados bastante inferiores al resto.

Media de aciertos en 10 iteraciones



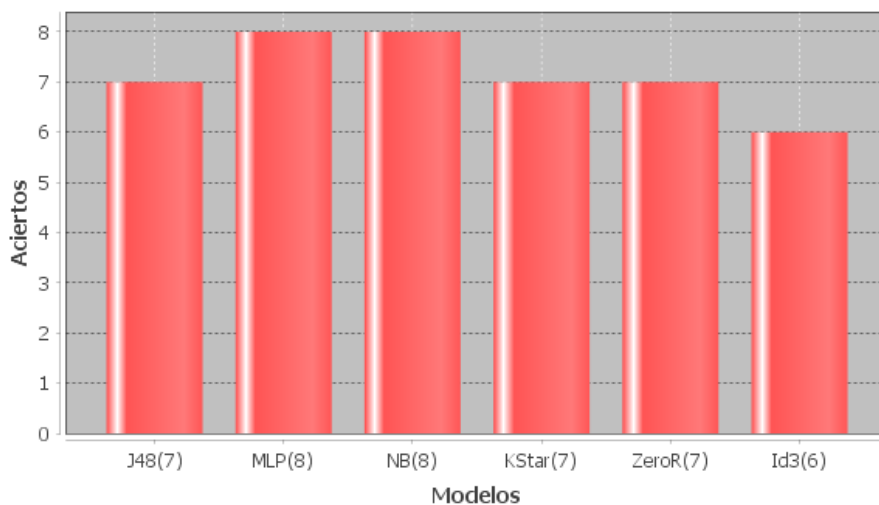
Conjunto de datos: arqueologia.arff

Total de instancias: 124.

- Partición del 80%:

Para este conjunto de datos los mejores modelos son Multilayer Perceptron y Naive Bayes. En este caso, el peor modelo es Id3 a diferencia de otros conjuntos donde ZeroR siempre era el peor modelo de entrenamiento. J48, KStar y ZeroR presentan los mismos resultados quedando así en segundo lugar.

Media de aciertos en 10 iteraciones

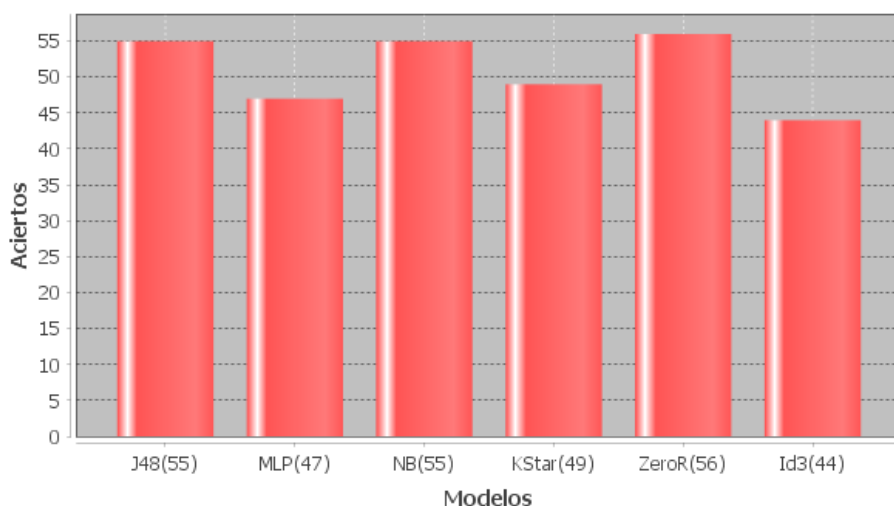


- Partición del 50%:

Para esta partición se presentan variaciones en los resultados. Estas variaciones pueden ser debidas a que este conjunto de datos tiene muy pocas instancias, por tanto, cuanto más pequeño es el conjunto de datos, menos fiables son los resultados.

El mejor modelo es ZeroR seguido del J48 y Naive Bayes. El peor modelo es Id3. Es sorprendente que para este conjunto de datos sea el modelo ZeroR el que presenta mejores resultados. De aquí se podría sacar la conclusión de que ZeroR funciona mejor con conjunto de datos más reducidos.

Media de aciertos en 10 iteraciones

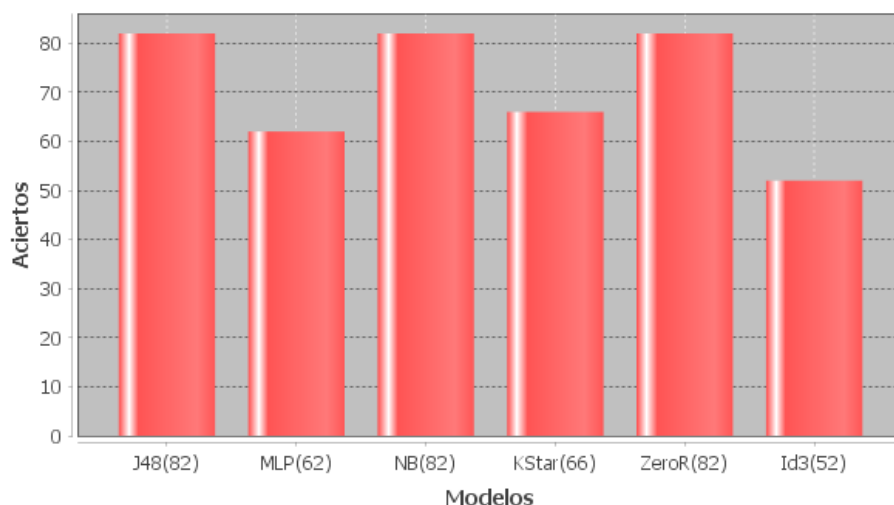


- Partición del 20%:

De forma proporcional, con un 20% de los datos para realizar el entrenamiento se obtienen los mismos resultados que con particiones mayores. J48, Naive Bayes y ZeroR son los mejores modelos para este conjunto, seguido de KStar y MLP. El peor modelo es Id3.

J48 e Id3 son modelos de entrenamiento de árboles de decisión, pero en cambio, para este conjunto de datos presentan resultados muy diferentes. Se comprueba que el J48 es el mejor modelo de entrenamiento de árboles de decisión.

Media de aciertos en 10 iteraciones



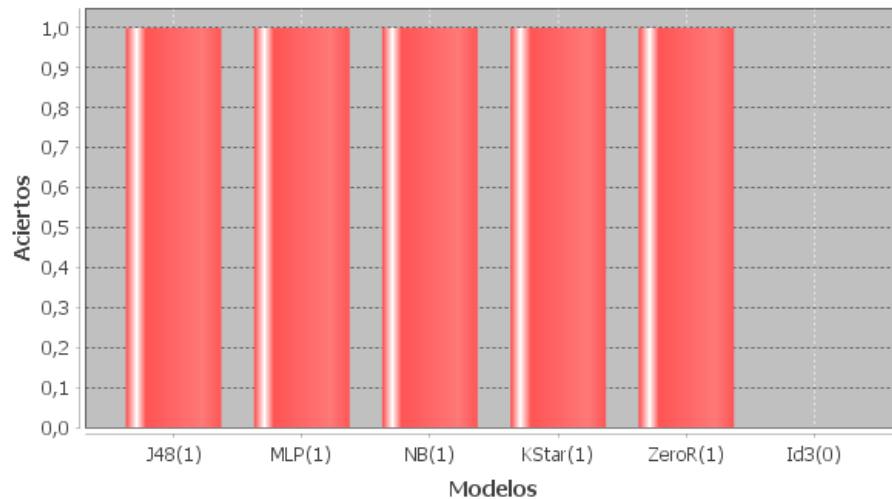
Conjunto de datos: weather.arff

Total de instancias: 14.

- Partición del 80%:

Este conjunto de datos es muy reducido con un total de 14 instancias. Por tanto, los resultados no pueden ser muy fiables. Con un 80% de los datos para el entrenamiento, no quedan muchos datos para realizar las pruebas de validación y de esta manera se presentan los siguientes resultados. Todos los modelos excepto el Id3 tienen de media 1 acierto y en cambio Id3 es el peor modelo para este conjunto con 0 aciertos de media.

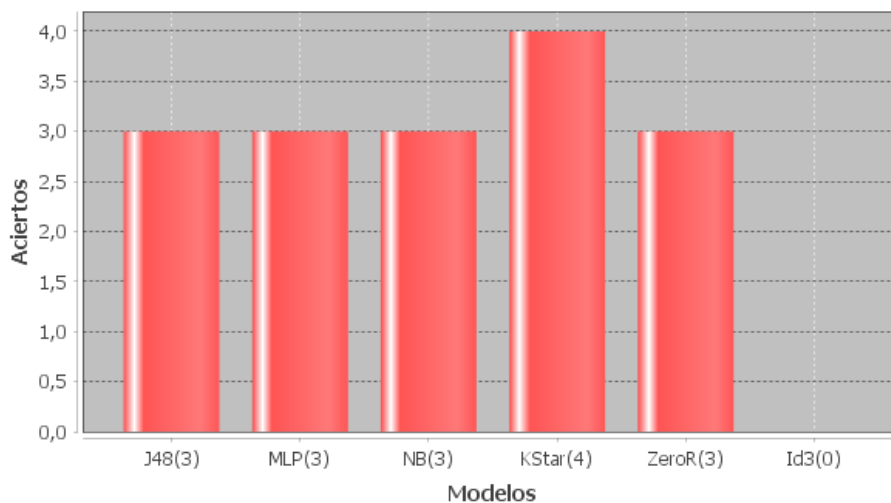
Media de aciertos en 10 iteraciones



- Partición del 50%:

KStar destaca entre los demás modelos de entrenamiento con 4 aciertos de media. Id3 sigue sin aumentar su media de aciertos quedando así en 0 y siendo el peor modelo. El resto de modelos están a la misma altura con 3 aciertos.

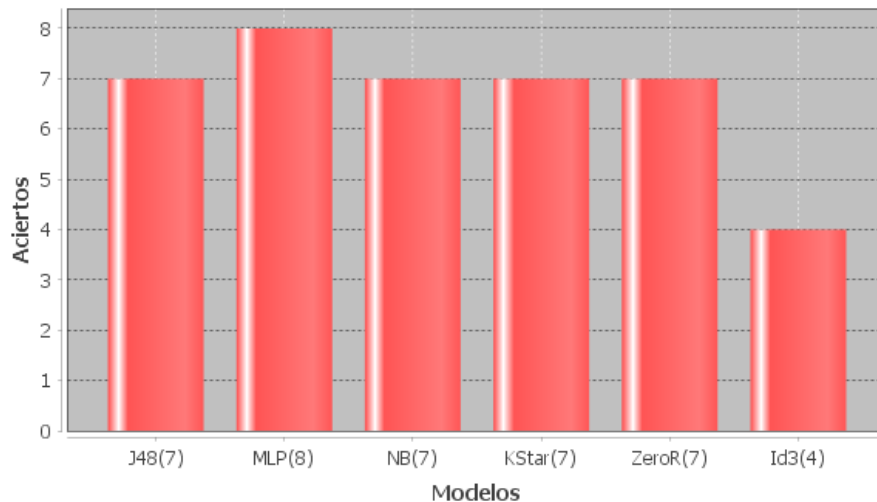
Media de aciertos en 10 iteraciones



- Partición del 20%:

Para esta partición, el mejor modelo ya no es KStar, sino MLP. Id3 ya obtiene algún acierto y el resto de modelos están a la misma altura con 7 aciertos de media. De aquí podríamos sacar la conclusión de que el modelo Id3 no funciona muy bien con conjuntos de datos muy pequeños y que los resultados pueden variar mucho y no ser muy fiables.

Media de aciertos en 10 iteraciones



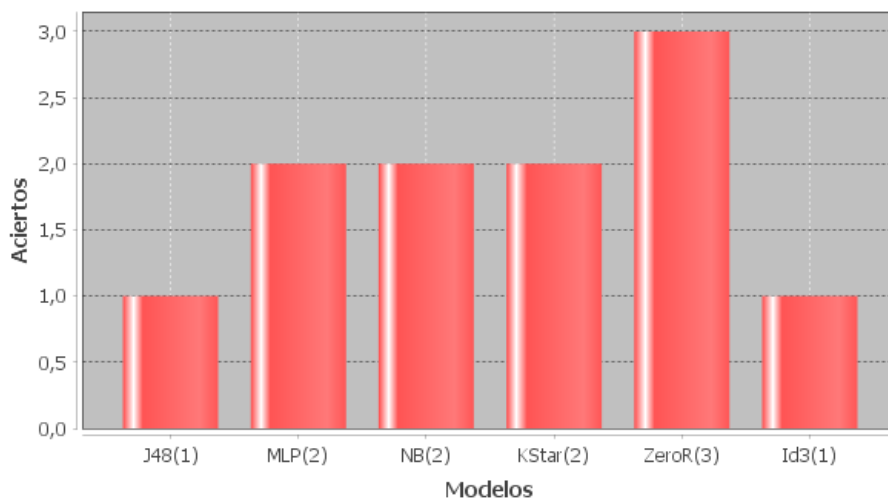
Conjunto de datos: empleados.arff

Total de instancias: 15.

- Partición del 80%:

Finalmente, con el conjunto de datos empleados se presentan los siguientes resultados: el mejor modelo de entrenamiento es el ZeroR, seguido de MLP, Naive Bayes y Kstar. Los peores modelos son los que utilizan árboles de decisión (J48 y Id3) quedando en último lugar con un acierto.

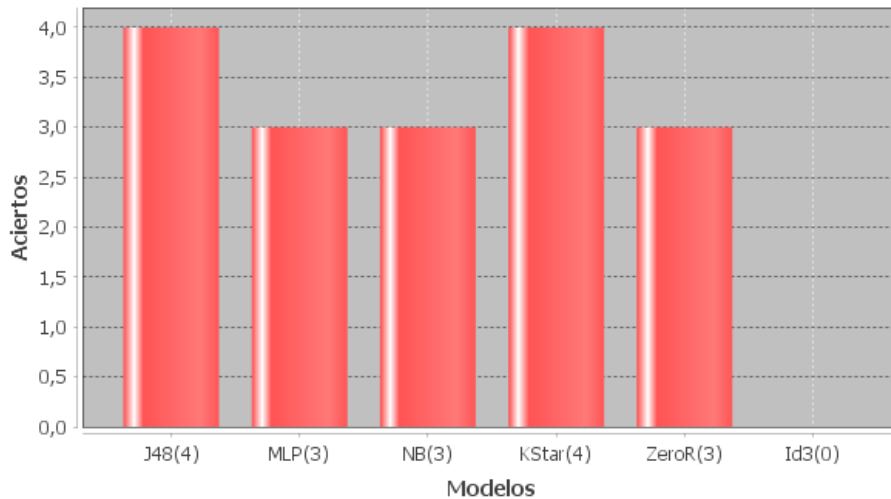
Media de aciertos en 10 iteraciones



- Partición del 50%:

Para la misma cantidad de datos para entrenamiento y para test presenta una gran mejoría el algoritmo J48 quedando en primer lugar con KStar. Id3 no obtiene ningún acierto y el resto de modelos obtienen los mismos resultados.

Media de aciertos en 10 iteraciones

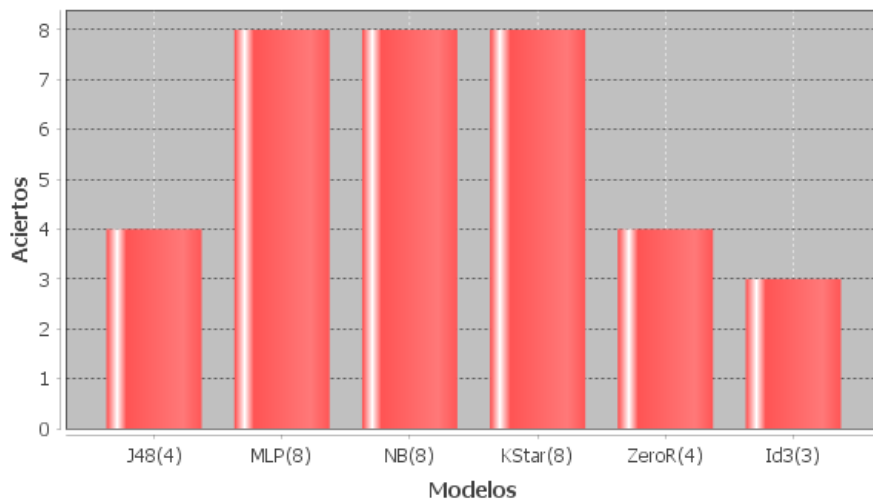


- Partición del 20%:

Para una partición de un 20% de instancias utilizadas para entrenamiento en este conjunto de datos se obtiene que el mejor resultado sea para MLP, Naive Bayes y KStar. J48 y ZeroR obtienen los mismos resultados y finalmente Id3 es el peor modelo para este dataset.

Como conclusión final, este conjunto de datos con 15 instancias hace muy difícil que un modelo de datos destaque por gran diferencia ante los demás. Según el conjunto de datos utilizado para el entrenamiento pueden variar mucho los resultados aunque se realicen 10 iteraciones.

Media de aciertos en 10 iteraciones



Gestión del proyecto

Organización del trabajo

Dadas las fechas de realización del trabajo (vacaciones de Navidad) y el número de personas en el grupo, se decidió que la mejor manera de organizar el trabajo era dividirnos en dos grupos en función de las tareas a realizar.

Por un lado estaba el *Equipo Knime* formado por Guillermo Pérez y Sandra Campos y por otro el *Equipo JADE* formado por Alejandro Bean y Andrea Aleixendri.

Se dedicó una jornada, como se explica en el diagrama GANTT más adelante, a organizar fechas y distribución del trabajo. Se llegó a la conclusión de que el *Equipo Knime* debería empezar antes para que el *Equipo JADE* pudiera contar con los workflows generados y poder inspirarse en ellos. No obstante, como se ha comentado anteriormente, durante la realización de la parte de JADE se descubrió que el algoritmo Apriori utilizado en Knime no era muy adecuado para comparar con los otros modelos dado que los resultados que presenta este modelo no muestra el número de instancias clasificadas correctamente. Por ello, hubo que dedicar un tiempo al final del proyecto a remodelar la parte de Knime para que estuviera en consonancia con la otra.

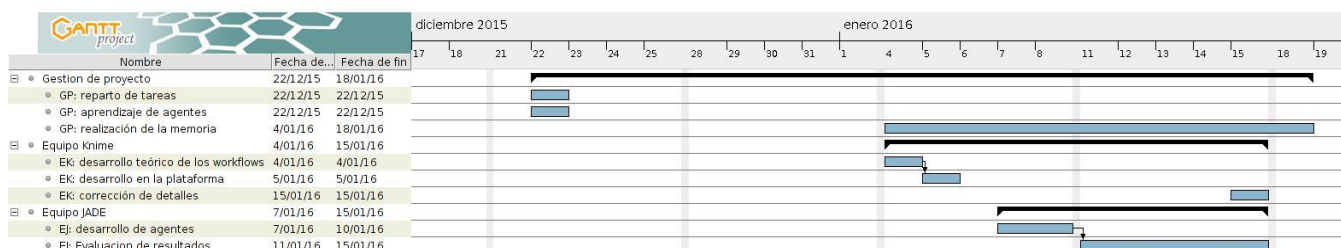
Dividiendo el trabajo en dos grupos diferenciados se ha conseguido focalizar y centrar más las tareas, haciéndolas más dinámicas y más fáciles de realizar. Además, dadas las nuevas tecnologías, la comunicación entre los dos subgrupos ha sido rápida y fluida, por lo que hemos quedado altamente satisfechos.

Herramientas utilizadas

Las herramientas con las que se ha trabajado durante la realización de este proyecto son:

- **JADE** (Java Agent Development Framework) es una herramienta de software libre programada en java en el año 2000 y que permite desarrollar sistemas multi-agente bajo el estándar FIPA^[1].
- **KNIME** (Konstanz Information Miner) es una plataforma de minería de datos, creada en el año 2011, que permite el desarrollo de modelos en un entorno visual. Está construido bajo la plataforma Eclipse.
- **WEKA** (*Waikato Environment for Knowledge Analysis*) es una plataforma de software para el aprendizaje automático y la minería de datos escrito en Java y desarrollado en la Universidad de Waikato. Weka es software libre distribuido bajo la licencia GNU-GPL.
- **Netbeans** es la plataforma de desarrollo que hemos utilizado para el desarrollo con JADE y la conexión con Weka a través de librerías.

Diagrama GANTT



Conclusiones

Valoración colectiva

Una vez finalizado el proyecto se ha realizado una valoración global del trabajo entre los cuatro miembros del equipo y el resultado ha sido bastante positivo en todos los aspectos.

Un punto que nos preocupaba al comenzar el trabajo era la organización del trabajo y la coordinación entre los miembros, que podía suponer un problema, ya que entre los distintos miembros del equipo teníamos horarios muy dispares: algunos trabajando, otros al estar realizando el trabajo de fin de grado, asignaturas de otros cursos, etc. Esto sumado a las fechas de realización del trabajo (vacaciones de Navidad) nos obligó a reunirnos para dividir el trabajo en dos equipos: uno que se encargaría de la parte de Knime y otro en la parte de JADE (se explica con más detalle en el punto de gestión de proyecto).

Tras finalizar el proyecto la valoración de este punto ha sido excelente, ya que a pesar de dividir el trabajo en dos equipos se ha mantenido en todo momento la comunicación entre todos, lo que ha facilitado que se logren los objetivos fijados al comienzo del proyecto.

La valoración de los resultados obtenidos ha sido notable: se ha cumplido con todas las tareas mínimas del trabajo establecidas además de dos tareas opcionales: se han probado tres modelos y dos datasets adicionales.

Como punto a mejorar, no se logró terminar la tercera tarea opcional que consistía en devolver los resultados de los agentes en formato CSV.

Para finalizar, nuestra valoración de la labor del profesorado respecto a este trabajo es muy buena.

Con los conocimientos dados durante el curso, tanto de clases teóricas como sesiones de prácticas y de programación de agentes, se cubrían todos los conocimientos mínimos para poder realizar un buen proyecto.

También ha facilitado la realización del proyecto el material ofrecido para el trabajo (código Agentes) y la ayuda ofrecida en las últimas sesiones teóricas de la asignatura.

Valoraciones individuales

- **Sandra:** Me ha parecido un trabajo bastante completo para iniciar nuestra andadura en el trabajo con sistemas agentes. Si bien al principio todo el tema de JADE y el trabajo con agentes en general imponía, al final considero que hemos sabido desenvolvemos bien en el trabajo propuesto.
- **Guillermo:** Para mí ha sido muy positivo que parte del trabajo requerido era igual al realizado en anteriores prácticas con la herramienta Knime, puesto que en los trabajos finales de otras asignaturas se piden cosas relacionadas con el temario pero que no se han dado en los 3 meses anteriores. No obstante también era necesario dar un paso más y aprender programación de agentes. También ha destacado a mi parecer el modo de evaluación, con unas tareas muy concretas, bien especificadas y con opción de obtener un extra de 2 puntos.
- **Alejandro:** Trabajo práctico interesante para asentar los conocimientos obtenidos en las sesiones teóricas de la asignatura. Personalmente satisfecho con haber aprendido programación de agentes y distintos algoritmos para minería de datos (Los escogidos y algunos que finalmente fueron rechazados).
- **Andrea:** ha sido un trabajo bastante interesante en el que hemos aprendido a programar agentes en Java y a utilizar varios modelos de datos. Poder utilizar Weka en Java a través de su librería con las mismas funciones que desde la aplicación de Weka de escritorio me parece una opción muy buena. La programación con agentes parecía que iba a ser una tarea bastante costosa y a la que habría que dedicar muchas horas pero el proporcionarnos la estructura y un ejemplo que pudiera servirnos para hacer pruebas con él nos ha servido

de mucha ayuda.

Bibliografía y enlaces de interés

[1] *Foundation for Intelligent Physical Agents* <http://www.fipa.org/>

[2] *Top 10 algorithms in data mining* <http://www.cs.umd.edu/~samir/498/10Algorithms-08.pdf>

[3] *An instance-based learner using an Entropic Distance Measure*

<http://www.cms.waikato.ac.nz/~ml/publications/1995/Cleary95-KStar.pdf>

[4] *Tutorial de desarrollo de sistemas multiagentes JADE. Jose Angel Bañares Universidad de Zaragoza.*

<http://webdiis.unizar.es/asignaturas/ISBC/lecciones/11.JADE.pdf>

[5] *API métodos Weka.*

<http://weka.sourceforge.net/doc.dev/>

Código Java y Knime disponible en <https://github.com/guillepg/SATD/>