PROYECTO FIN DE CARRERA

PLATAFORMA DE EVALUACIÓN DE ALGORITMOS PARA LA IDENTIFICACIÓN DE USUARIOS

ALHARETH DAWOOD

UNIVERSIDAD POLITÉCNICA DE MADRID

Escuela Técnica Superior de Ingeniería y Sistemas de Telecomunicación

E.T.S.I.S. TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA

PLAN 2000

TEMA: Aprendizaje automático e inteligencia ambiental

TITULO: Plataforma de evaluación de algoritmos para la identificación de usuarios.

AUTOR: Alhareth dawood

TUTOR: Iván Pau de la Cruz

DEPARTAMENTO: DIATEL **Vº Bº**

Miembros del Tribunal Calificador:

PRESIDENTE: Pedro José Lobo Perea

VOCAL: Iván Pau de la Cruz

VOCAL SECRETARIO: Ana Belén García Hernando

DIRECTOR:

Fecha de lectura:

Calificación: **El Secretario:**

RESUMEN DEL PROYECTO

El proyecto planteado está definido para la creación de una plataforma que permita evaluar algoritmos de aprendizaje automático como mecanismos de identificación en espacios inteligentes.

Se estudiarán tanto los algoritmos propios de este tipo de técnicas como las plataformas actuales existentes para definir un conjunto de requisitos específicos de la plataforma a desarrollar.

Tras el análisis se desarrollará parcialmente la plataforma. Tras el desarrollo se validará con pruebas de concepto y finalmente se verificará en un entorno de investigación a definir.

Nota de aceptación

_____

___Este trabajo cumple con la calidad _____

___mínima exigida y necesaria para ser___

___presentado como Proyecto Fin de _____

___Carrera por el estudiante_____

Firma del Tutor

_____

Firma del Cotutor

_____

_____

Presidente del Jurado

_____

Jurado

_____

Jurado

A mi padre, mis hermanos, mi hermana, y en especial a Nawal mi madre esa gran mujer que ha sido mi luz en estos largos años, y a mi abuela por su sabiduría que en paz descanse.

AGRADECIMIENTOS

Quisiera agradecer a todas esas personas que me han apoyado tanto durante este tiempo. En primer lugar quisiera agradecer a mi tutor de proyecto Iván Pau de la Cruz su ayuda y apoyo sin los cuales habría sido imposible realizar este proyecto. Por último no me quiero olvidar de mis compañeros de facultad, familia y amigos, que gracias a ellos todo es siempre mucho más fácil.

Resumen

La minería de datos es un campo de las ciencias de la computación referido al proceso que intenta descubrir patrones en grandes volúmenes de datos. La minería de datos busca generar información similar a la que podría producir un experto humano. Además es el proceso de descubrir conocimientos interesantes, como patrones, asociaciones, cambios, anomalías y estructuras significativas a partir de grandes cantidades de datos almacenadas en bases de datos, data warehouses o cualquier otro medio de almacenamiento de información.

El aprendizaje automático o aprendizaje de máquinas es una rama de la Inteligencia artificial cuyo objetivo es desarrollar técnicas que permitan a las computadoras aprender. De forma más concreta, se trata de crear programas capaces de generalizar comportamientos a partir de una información no estructurada suministrada en forma de ejemplos. La minería de datos utiliza métodos de aprendizaje automático para descubrir y enumerar patrones presentes en los datos.

En los últimos años se han aplicado las técnicas de clasificación y aprendizaje automático en un número elevado de ámbitos como el sanitario, comercial o de seguridad. Un ejemplo muy actual es la detección de comportamientos y transacciones fraudulentas en bancos. Una aplicación de interés es el uso de las técnicas desarrolladas para la detección de comportamientos fraudulentos en la identificación de usuarios existentes en el interior de entornos inteligentes sin necesidad de realizar un proceso de autenticación.

Para comprobar que estas técnicas son efectivas durante la fase de análisis de una determinada solución, es necesario crear una plataforma que de soporte al desarrollo, validación y evaluación de algoritmos de aprendizaje y clasificación en los entornos de aplicación bajo estudio.

El proyecto planteado está definido para la creación de una plataforma que permita evaluar algoritmos de aprendizaje automático como mecanismos de identificación en espacios inteligentes. Se estudiarán tanto los algoritmos propios de este tipo de técnicas como las plataformas actuales existentes para definir un conjunto de requisitos específicos de la plataforma a desarrollar. Tras el análisis se desarrollará parcialmente la plataforma. Tras el desarrollo se validará con pruebas de concepto y finalmente se verificará en un entorno de investigación a definir.

Abstract

The data mining is a field of the sciences of the computation referred to the process that it tries to discover patterns in big volumes of information. The data mining seeks to generate information similar to the one that a human expert might produce. In addition it is the process of discovering interesting knowledge, as patterns, associations, changes, abnormalities and significant structures from big quantities of information stored in databases, data warehouses or any other way of storage of information.

The machine learning is a branch of the artificial Intelligence which aim is to develop technologies that they allow the computers to learn. More specifically, it is a question of creating programs capable of generalizing behaviors from not structured information supplied in the form of examples. The data mining uses methods of machine learning to discover and to enumerate present patterns in the information.

In the last years there have been applied classification and machine learning techniques in a high number of areas such as healthcare, commercial or security. A very current example is the detection of behaviors and fraudulent transactions in banks. An application of interest is the use of the techniques developed for the detection of fraudulent behaviors in the identification of existing Users inside intelligent environments without need to realize a process of authentication.

To verify these techniques are effective during the phase of analysis of a certain solution, it is necessary to create a platform that support the development, validation and evaluation of algorithms of learning and classification in the environments of application under study.

The project proposed is defined for the creation of a platform that allows evaluating algorithms of machine learning as mechanisms of identification in intelligent spaces. There will be studied both the own algorithms of this type of technologies and the current existing platforms to define a set of specific requirements of the platform to develop. After the analysis the platform will develop partially. After the development it will be validated by prove of concept and finally verified in an environment of investigation that would be define.

Tabla de Contenidos

Lista de Tablas

Lista de Figuras

Capítulo 1

1. Descripción del proyecto

La Minería de Datos (Data Mining) es la búsqueda de patrones y de regularidades destacables en grandes bases de datos. El Aprendizaje Automático es el campo de la Ingeniería Informática en el que se estudian y desarrollan algoritmos que implementan los distintos modelos de aprendizaje y su aplicación a la resolución de problemas prácticos. La minería de datos utiliza métodos de aprendizaje automático para descubrir y enumerar patrones presentes en los datos.

En los últimos años se han aplicado las técnicas de clasificación y aprendizaje automático en un número elevado de ámbitos como el sanitario, comercial o de seguridad. Un ejemplo muy actual es la detección de comportamientos y transacciones fraudulentas en bancos. Una aplicación de interés es el uso de las técnicas desarrolladas para la detección de comportamientos fraudulentos en la identificación de usuarios existentes en el interior de entornos inteligentes sin necesidad de realizar un proceso de autenticación.

Para comprobar que estas técnicas son efectivas durante la fase de análisis de una determinada solución, es necesario crear una plataforma que de soporte al desarrollo, validación y evaluación de algoritmos de aprendizaje y clasificación en los entornos de aplicación bajo estudio.

1.1 Objetivos del proyecto

1.1.1 Objetivo Principal

El PFC tiene como principal objetivo diseñar e implementar una plataforma que facilite la tarea de verificación de algoritmos de aprendizaje y descubrimiento automático de patrones en un entorno propio de la minería de datos. La plataforma, además de incluir las funciones necesarias para la ejecución y evaluación de los algoritmos, debe permitir el acceso múltiple y la automatización de los procesos de evaluación de los algoritmos para un conjunto dado de entradas.La plataforma, una vez desarrollada, se deberá configurar para facilitar el proceso de identificación de usuarios en espacios inteligentes.

1.1.2 Objetivos Específicos

- Análisis y comprensión del funcionamiento de las técnicas de minería de datos con el fin de encontrar la que más se adapta a nuestras necesidades y a los datos analizados para conseguir un resultado óptimo.
- Análisis y comprensión del funcionamiento de los principales Algoritmos de Aprendizaje automático y usar cada algoritmo según sea la situación a analizar.
- Análisis e investigación de las tecnologías actuales e implementar las correspondientes para llegar a realizar nuestro objetivo.
- Definición de requisitos y diseño general de la arquitectura y de cada uno de los componentes implicados en dicha arquitectura.
- Desarrollar una plataforma que cumpla con los requisitos funcionales que comprenden el comportamiento interno del software: cálculos, detalles, técnicas, manipulación de datos, así como los no funcionales que se enfocan en el diseño y la implementación de la plataforma.

1.1.3 Alcance

La construcción de la plataforma completa es un trabajo que excede el esfuerzo asignado al presente PFC. Por lo tanto el alcance queda limitado al desarrollo de la plataforma en una arquitectura que permita posteriormente cumplir el resto de los requisitos. En concreto no será parte de los objetivos operativos la automatización de los procesos de evaluación de los algoritmos.

Además en el presente PFC no se configura la plataforma resultante con el objetivo específico de la identificación de usuarios. La plataforma queda definida como una herramienta genérica quedando para posteriores desarrollos el afinamiento necesario para incluirla en un entorno como el citado.

1.2 Metodología

La metodología para el desarrollo del PFC, se ha centrado en el modelo de Desarrollo Incremental, cuyo ciclo de vida se basa en la mejora iterativa para desarrollar un sistema de manera incremental, permitiéndole al desarrollador aprovechar las ventajas de lo aprendido en los ciclos anteriores y aplicarlo en el siguiente. Al tratarse de un PFC, creemos que es la metodología adecuada para asimilar los conceptos que serán desarrollados durante el proyecto.

Los pasos claves en el proceso se basan en comenzar con una implementación simple de los requisitos del sistema, e iterativamente mejorar la secuencia evolutiva de versiones hasta que el sistema completo esté implementado. En cada iteración, se realizan cambios en el diseño y se agregan nuevas funcionalidades y capacidades al sistema. A nivel pedagógico, permite identificar los errores cometidos y reforzar los conceptos aprendidos.

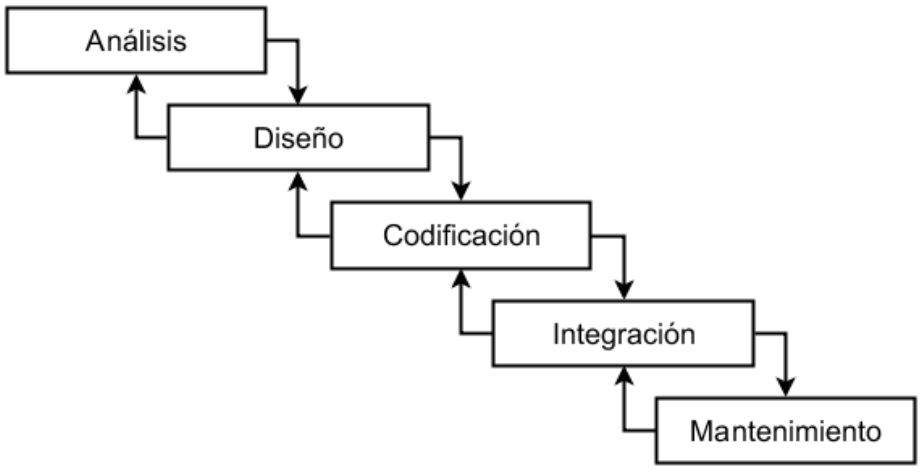


Figura 1 - Desarrollo Incremental

1.3 Estructura del documento

Esta memoria sigue la siguiente estructura:

Capítulo 1. Descripción del proyecto: En esta sección o capítulo se da una prevé descripción del proyecto y de los objetivos tanto principales como específicos y los pasos a implementar para llegar al cumplimiento de dichos objetivos.

Capítulo 2. Extracción del conocimiento y Minería de datos: En este capítulo o sección presentamos un análisis amplio del proceso que hay que seguir para aplicar los modelos de minerías de datos sobre un conjunto o estructura de datos y la evaluación de los resultados obtenidos. Además de analizar los modelos de minería de datos que se usan hoy en día.

Capítulo 3. Algoritmos de Aprendizaje automático. En éste capítulo se estudian los diferentes algoritmos que se han utilizado para el desarrollo de la aplicación.

Capítulo 4. Arquitectura y diseño: En éste capítulo se describe la arquitectura en la que se apoya la aplicación, así como el diseño del prototipo resultante de este proyecto.

Capítulo 5. Pruebas: Contiene una serie de pruebas para comprobar el correcto funcionamiento de la aplicación. Se presentan los resultados y la viabilidad de la propuesta.

Capítulo 6. Conclusiones: En éste capítulo se presenta un resumen de las aportaciones de este PFC y los resultados obtenidos. Así como el trabajo que a corto y medio plazo que se puede realizar para extenderlo y tratar de generar algún aporte a la comunidad académica.

Capítulo 2

2. Extracción del conocimiento y minería de Datos (1)

El proceso de extracción del conocimiento a partir de datos ("Knowledge Discovery in Databases" o KDD) incorpora diferentes técnicas (arboles de decisión, regresión lineal, redes neuronales artificiales, técnicas bayesianas, maquinas de soporte vectorial, etc.) de diversos campos (aprendizaje automático e inteligencia artificial, estadística, base de datos, etc.) y aborda una tipología variada de problemas (clasificación, categorización, estimación, regresión, agrupamiento, etc.).

El objetivo general del proceso de extracción del conocimiento consiste en extraer información de un conjunto de datos y transformarla en una estructura comprensible para su uso posterior.

La minería de datos (es la etapa de análisis de KDD según [ODE10]), es un campo de las ciencias de la computación referido al proceso que intenta descubrir patrones en grandes volúmenes de datos. La minería de datos busca generar información similar a la que podría producir un experto humano.

(1) Parte importante de la información de este capítulo ha sido recogida de las siguientes fuentes [MON07] [MIT97]

Además es el proceso de descubrir conocimientos interesantes, como patrones, asociaciones, cambios, anomalías y estructuras significativas a partir de grandes cantidades de datos almacenadas en bases de datos, data warehouses o cualquier otro medio de almacenamiento de información.

En resumen el *Data Mining* (DM) es un conjunto de técnicas de análisis de datos que permiten:

- Extraer **Patrones, Tendencias y Regularidades** para describir y comprender mejor los datos.
- Extraer **Patrones y Tendencias** para predecir comportamientos futuros.

La idea de *Minería de Datos* no es nueva. Ya desde los años sesenta los estadísticos manejaban términos como *Data Fishing*, *Data Mining (DM) o Data Archaeology* con la idea de encontrar correlaciones sin una hipótesis previa en *bases de datos* con ruido.

Esta tecnología ha sido un buen punto de encuentro entre personas pertenecientes al ámbito académico y al de los negocios. La evolución de sus herramientas en el transcurso del tiempo puede dividirse en cuatro etapas principales:

- Colección de Datos (década de los 60).
- Acceso de Datos (década de los 80).
- Almacén de Datos y Apoyo a las Decisiones (principios de la década de 1990).
- Minería de Datos Inteligente. (finales de la década de 1990).

2.1 Etapas del proceso de minería de datos

La tarea de minería de datos real es el análisis automático o semiautomático de grandes cantidades de datos para extraer patrones interesantes hasta ahora desconocidos, como los grupos de registros de datos (análisis clúster), registros poco usuales (la detección de anomalías) y dependencias.

El proceso de minería de datos que vamos a seguir es el siguiente:

- **Selección de datos:** definir el origen de datos y transformación del conjunto de datos de entrada.
- **Preprocesado y transformación de datos:** el objetivo de este paso es preparar y procesar los datos para aplicar la técnica de minería de datos que mejor se adapte a los datos y al problema.
- **Técnicas de modelado:** en esta etapa se construye el modelo predictivo o descriptivo, se decide que algoritmo de aprendizaje automático o que técnica estadística escoger. Se

obtiene un modelo de conocimiento que representa patrones de comportamiento observados en los valores de las variables del problema o relaciones de asociación entre dichas variables. También pueden usarse varias técnicas o algoritmos a la vez para generar distintos modelos, aunque generalmente cada técnica o algoritmo obliga a un preprocesado diferente de los datos.

- **Presentación de resultados (Interpretación y evaluación de datos):** técnicas de visualización y de representación del conocimiento, una vez obtenido el modelo, se debe proceder a su validación comprobando que las conclusiones que arroja son válidas y suficientemente satisfactorias. En el caso de haber obtenido varios modelos mediante el uso de distintas técnicas, se deben comparar los modelos en busca de aquel que se ajuste mejor al problema. Si ninguno de los modelos alcanza los resultados esperados debe alterarse alguno de los pasos anteriores para generar nuevos modelos.

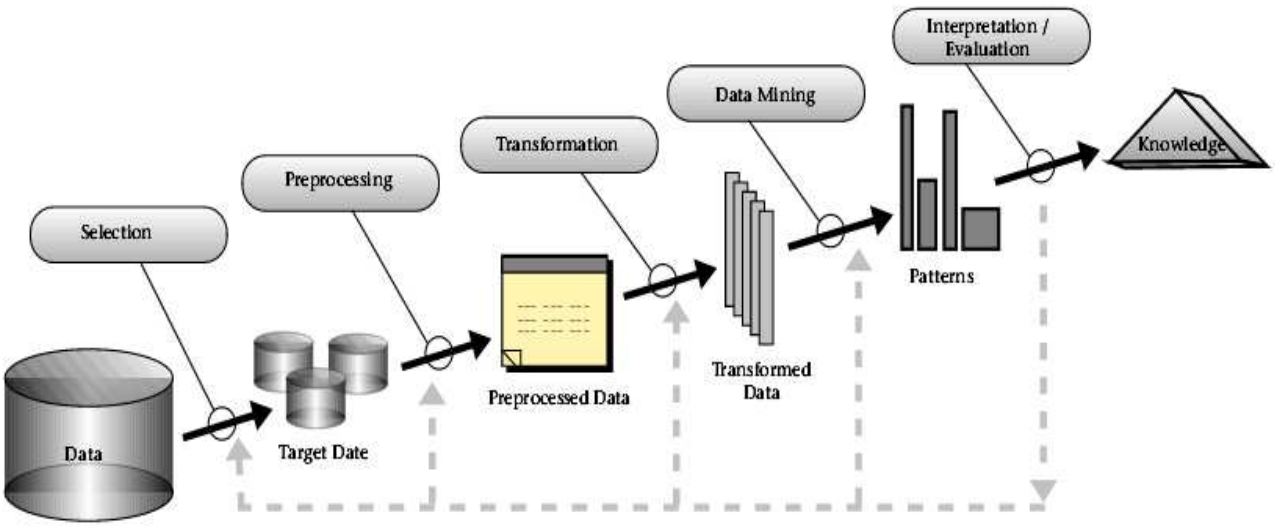


Figura 2 - Etapas del proceso de minería de datos

Fuente [FAY96]

2.2 Selección de datos

En este primer paso definimos el origen de los datos, podríamos disponer de varias fuentes de datos como pueden ser:

- Un fichero que contiene los atributos y los datos a procesar.
- Conectando con una base de datos.

En el caso de que el origen de datos sea un fichero que contiene los datos a procesar. Tenemos que definir una estructura para dicho fichero. Para tomar ese fichero como valido su estructura tiene que cumplir las condiciones de la estructura definida.

Podemos definir una estructura de la siguiente forma:

- Sección de cabecera.
- Sección de atributos.
- Sección de datos.

Vemos que aparte de la sección de datos tenemos otras dos secciones más, que tendrán meta-información sobre los propios datos.

En la sección de atributos además de indicar el nombre del atributo tenemos que indicar el tipo de dicho atributo (número entero, cadena de caracteres, fecha, etc.). Se puede hacer con una descripción textual del tipo de dato o también podemos tomar como referencia la estructura de tipos de datos de java.

Además habrá que definir una extensión para dicho fichero tal que los ficheros que no tengan esa extensión no se tomaran como validos. Otra posible opción interesante seria admitir archivos CSV por ejemplo, tal que la primera línea sea los atributos de los datos a procesar. Esto lo veremos en los apartados correspondientes en la fase de diseño de la plataforma.

En el caso de optar por un origen de datos a través de una conexión de base de datos, tenemos que tener en cuenta que al usar java pues haremos dicha conexión con un driver JDBC.

2.3 Preprocesado y transformación de datos

El formato de los datos contenidos en la fuente de datos (base de datos, Data Warehouse) usualmente no es el idóneo y la mayoría de las veces no es posible extraer información de los datos "en bruto".

Mediante el preprocesado se filtran los datos (de forma que se eliminan valores incorrectos, no válidos, desconocidos... según las necesidades y el algoritmo que va a usarse).

En cuanto los datos estén cargados en la plataforma de forma correcta, presentamos cada uno de los atributos que componen los datos. Esa presentación del conjunto de atributos y datos nos tiene que dar una información detallada de toda la estructura de datos que acabamos de cargar. Además nos dará un resumen estadístico de los mismos (media aritmética, rango de los datos, desviación estándar, número de instancias distintas, de qué tipo son, etc.).

Sería interesante también tener esa información detallada junto el resumen estadístico de una forma grafica. Así tenemos los datos representados de forma textual y grafica antes de aplicar los algoritmos de aprendizaje automático.

Una parte importante de la preparación de datos es poder filtrar los datos aplicando diversos filtros. Algunos ejemplos de filtros sobre los datos serian añadir atributos, quitar atributos, añadir atributos que son una función de los atributos existentes, etc. De esta forma ya tendríamos los datos preparados y filtrados.

2.4 Técnicas de modelado

Para crear un modelo de minería de datos, primero se analiza un conjunto de datos buscando patrones y tendencias específicas. Después, se utilizan los resultados de este análisis para definir los parámetros del modelo de minería de datos.

Un modelo de minería de datos puede tomar diversas formas. Algunos ejemplos son:

- Un conjunto de reglas que describen cómo se agrupan los productos en una transacción.
- Un árbol de decisión que predice si un cliente determinado comprará un producto.
- Un modelo matemático que predice las ventas.
- Un conjunto de clústeres que describe cómo se relacionan los escenarios de un conjunto de datos.

Los modelos de minería de datos se clasifican en dos grandes categorías:

- Modelos predictivos.
- Modelos descriptivos.

Los modelos predictivos se utilizan para prever el comportamiento futuro de algún tipo de entidad mientras que los descriptivos pueden ayudar a su comprensión. En la figura 3 se muestra una clasificación de las distintas técnicas utilizadas en la minería de datos.

Las técnicas utilizadas en la minería de datos provienen de la inteligencia artificial, aprendizaje automático y de la estadística. Dichas técnicas no son más que algoritmos que se aplican sobre un conjunto de datos para obtener unos determinados resultados.

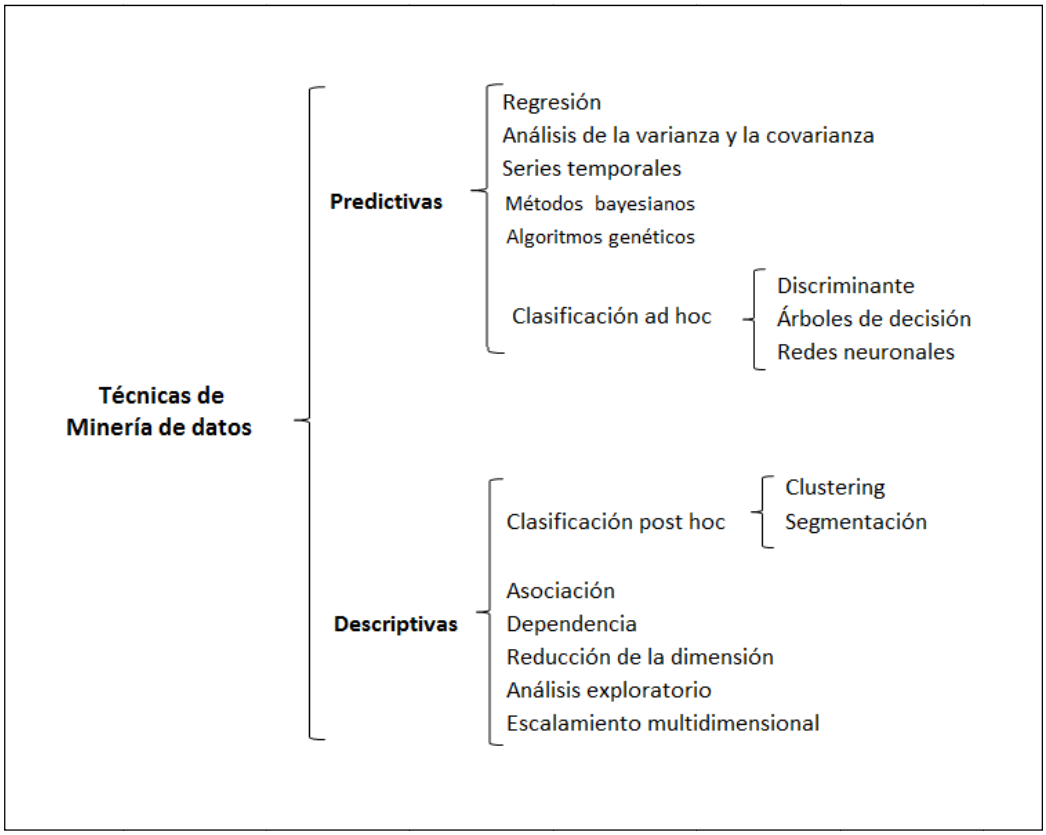


Figura 3 - Técnicas de minería de datos

Basado en Fuente [MON07]

2.4.1 Modelo predictivo

Como se comento anteriormente un modelo predictivo, trata de predecir o responder a preguntas futuras en base a un estudio de su comportamiento pasado. Provee información sobre estados futuros del sistema.

Algunas de las preguntas que podríamos responder con este tipo de modelo de datos son:

- ¿Qué tal se venderá el próximo año un determinado producto?
- ¿Dónde se producirá el siguiente atentado terrorista?
- ¿Qué riesgo tiene cierta persona de contraer una determinada enfermedad?
- ¿Qué clientes tienen más riesgos de darse de baja de nuestra empresa?

Los métodos predictivos se agrupan a grandes rasgos en las siguientes familias:

- Regresión.
- Análisis de la varianza y covarianza.
- Series temporales.
- Métodos bayesianos.
- Algoritmos genéticos.
- Clasificación ad-hoc discriminante.
- Clasificación ad-hoc árboles de decisión.
- Clasificación ad-hoc redes neuronales.

A continuación se detallan las características más relevantes de las familias de métodos enumerados.

Regresión

Define la relación entre una o más variables. La regresión se utiliza para predecir una medida basándonos en el conocimiento de otra. La regresión estadística o regresión a la media es la tendencia de una medición extrema a presentarse más cercana a la media en una segunda medición. La regresión se utiliza para predecir una medida basándonos en el conocimiento

de otra. La regresión trata de encontrar una fórmula que proporcione el valor objetivo dadas las variables de entrada. Por ejemplo, en el caso de una regresión lineal:

Y = w0 + x1*w1 + x2*w2 +… + xN*wN (siendo xn la variable y wn el peso)

El modelo de regresión requiere un mayor tratamiento de los datos, puesto que sólo trata con valores numéricos. Las categorías o clases hay que transformarlas en numéricos. No puede manejar valores nulos o perdidos. Además es necesario detectar y manejar los valores extremos o atípicos y hay que tener en cuenta las no linealidades en los datos. El origen de los valores perdidos puede ser variado: No ser una medida aplicable en el registro, proceder de una unión de tablas desemparejada, no haberse introducido el valor. Dependiendo del caso, se aplican diferentes remedios de reemplazamiento: métodos de distribución sintética, métodos de estimación.

Análisis de la varianza y covarianza

En estadística, el análisis de la varianza (ANOVA, ANalysis Of VAriance, según

terminología inglesa) es una colección de modelos estadísticos y sus procedimientos asociados, en el cual la varianza está particionada en ciertos componentes debidos a diferentes variables explicativas.

Las técnicas iniciales del análisis de varianza fueron desarrolladas por el estadístico y genetista R. A. Fisher en los años 1920 y 1930 y es algunas veces conocido como "Anova de Fisher" o "análisis de varianza de Fisher" según [SPI07], debido al uso de la distribución F de Fisher como parte del contraste de hipótesis.

El análisis de la covarianza o ANCOVA (acrónimo del inglés analysis of covariance) es un modelo lineal general con una variable cuantitativa y uno o más factores. El ANCOVA es una fusión de la ANOVA y de la regresión lineal múltiple. Es un procedimiento estadístico que permite eliminar la heterogeneidad causada en la variable de interés (variable dependiente) por la influencia de una o más variables cuantitativas (covariables).

Básicamente, el fundamento del ANCOVA es un ANOVA al que a la variable dependiente se le ha eliminado el efecto predicho por una o más covariables por regresión lineal múltiple. La inclusión de covariables puede aumentar la potencia estadística porque a menudo reduce la variabilidad.

Series temporales

Una serie temporal o cronológica es una secuencia de datos, observaciones o valores, medidos en determinados momentos del tiempo, ordenados cronológicamente y normalmente espaciados entre sí de manera uniforme. El análisis de series temporales comprende métodos que ayudan a interpretar este tipo de datos, extrayendo información representativa, tanto referente a los orígenes o relaciones subyacentes como a la posibilidad de extrapolar y predecir su comportamiento futuro. De hecho, uno de los usos más habituales de las series de datos temporales es su análisis para predicción y pronóstico. Por ejemplo de los datos climáticos de las acciones de bolsa, o las series pluviométricas. Resulta difícil imaginar una rama de las ciencias en la que no aparezcan datos que puedan ser considerados como series temporales. Son estudiadas en estadística, procesamiento de señales, econometría y muchas otras áreas.

Métodos bayesianos

En la teoría de la probabilidad el teorema de Bayes es un resultado enunciado por Thomas Bayes en 1763 según [BAY63], que expresa la probabilidad condicional de un evento aleatorio A dado B en términos de la distribución de probabilidad condicional del evento B dado A y la distribución de probabilidad marginal de sólo A.

Teorema de Bayes

Sea $\{A_1, A_2, ..., A_i, ..., A_n\}$ un conjunto de sucesos mutuamente excluyentes y exhaustivos, y tales que la probabilidad de cada uno de ellos es distinta de cero (0). Sea B un suceso cualquiera del que se conocen las probabilidades condicionales $P(B|A_i)$. Entonces, la probabilidad $P(A_i|B)$ viene dada por la expresión:

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{P(B)}$$

Donde:

- $P(A_i)$ son las probabilidades a priori.
- $P(B|A_i)$ es la probabilidad de B en la hipótesis A_i.
- $P(A_i|B)$ son las probabilidades a posteriori.

En términos más generales y menos matemáticos, el teorema de Bayes es de enorme relevancia puesto que vincula la probabilidad de A dado B con la probabilidad de B dado A. Es decir que sabiendo la probabilidad de tener un dolor de cabeza dado que se tiene gripe, se podría saber -si se tiene algún dato más-, la probabilidad de tener gripe si se tiene un dolor de cabeza.

Muestra este sencillo ejemplo la alta relevancia del teorema en cuestión para la ciencia en todas sus ramas, puesto que tiene vinculación íntima con la comprensión de la probabilidad de aspectos causales dados los efectos observados.

Los métodos bayesianos, basados en el conocido teorema de Bayes. Todos ellos tienen en común la asignación de una probabilidad como medida de credibilidad de las hipótesis. Matemáticamente se trata de obtener las probabilidades de las hipótesis condicionadas a las evidencias que se conocen.

La actualización de las probabilidades hipótesis condicionadas a las evidencias se fundamenta en la aplicación del Teorema de Bayes. La diferencia entre los distintos métodos bayesianos, modelos causales y redes bayesianas, estriba en las hipótesis de independencia condicional entre hipótesis y evidencias. Dichas relaciones se expresan comúnmente mediante un grafo acíclico dirigido.

Algoritmos genéticos

Los algoritmos genéticos, son una técnica matemática de búsqueda y optimización que encuentra soluciones a un problema basándose en los principios que rigen la evolución de las especies a nivel genético molecular. Estos algoritmos requieren de un conjunto de datos para realizar su proceso de aprendizaje.

Los algoritmos genéticos imitan la evolución de las especies mediante la mutación, reproducción y selección, como también proporcionan programas y optimizaciones que pueden ser usadas en la construcción y entrenamiento de otras estructuras como es el caso de las redes neuronales. Además los algoritmos genéticos son inspirados en el principio de la supervivencia de los más aptos.

En los años 1970, de la mano de John Henry Holland [DOM05], surgió una de las líneas más prometedoras de la inteligencia artificial, la de los algoritmos genéticos.

Son llamados así porque se inspiran en la evolución biológica y su base genético-molecular. Estos algoritmos hacen evolucionar una población de individuos sometiéndola a acciones aleatorias semejantes a las que actúan en la evolución biológica (mutaciones y

recombinaciones genéticas). Así como también una selección de acuerdo con algún criterio, en función del cual se decide cuáles son los individuos más adaptados, que sobreviven y cuáles los menos aptos que son descartados. Es incluido dentro de los algoritmos evolutivos, que incluyen también las estrategias evolutivas, la programación evolutiva y la programación genética. Dentro de esta última se han logrado avances curiosos:

Un algoritmo genético es un método de búsqueda dirigida basada en probabilidad. Bajo una condición muy débil (que el algoritmo mantenga elitismo, es decir, guarde siempre al mejor elemento de la población sin hacerle ningún cambio) se puede demostrar que el algoritmo converge en probabilidad al óptimo. En otras palabras, al aumentar el número de iteraciones, la probabilidad de tener el óptimo en la población tiende a 1.

Clasificación ad-hoc discriminante

El análisis discriminante se conoce en ocasiones como análisis de la clasificación, ya que su objetivo fundamental es producir una regla o un esquema de clasificación que permite a un investigador predecir la población a la que es más probable que tenga que pertenecer una nueva observación o individuo.

Se trata de una técnica estadística que permite asignar o clasificar nuevos individuos u observaciones dentro de grupos previamente definidos.

Clasificación ad-hoc árboles de decisión

Está técnica se encuentra dentro de una metodología de aprendizaje supervisado. Su representación es en forma de árbol en donde cada nodo es una decisión, los cuales a su vez generan reglas para la clasificación de un conjunto de datos. Un árbol de decisión es un modelo de predicción utilizado en el ámbito de la inteligencia artificial. Dada una base de datos se construyen diagramas de construcciones lógicas, muy similares a los sistemas de predicción basados en reglas, que sirven para representar y categorizar una serie de condiciones que ocurren de forma sucesiva, para la resolución de un problema. Los árboles de decisión son fáciles de usar, admiten atributos discretos y continuos, tratan bien los atributos no significativos y los valores faltantes. Su principal ventaja es la facilidad de interpretación.

En el diseño de aplicaciones informáticas, un árbol de decisión indica las acciones a realizar en función del valor de una o varias variables. Es una representación en forma de árbol cuyas ramas se bifurcan en función de los valores tomados por las variables y que terminan en una

acción concreta. Se suele utilizar cuando el número de condiciones no es muy grande (en tal caso, es mejor utilizar una tabla de decisión).

Clasificación ad-hoc redes neuronales

Esta técnica de inteligencia artificial, en los últimos años se ha convertido en uno de los instrumentos de uso frecuente para detectar categorías comunes en los datos, debido a que son capaces de detectar y aprender complejos patrones, y características de los datos. Las redes de neuronas artificiales (denominadas habitualmente como RNA o en inglés como: "ANN") son un paradigma de aprendizaje y procesamiento automático inspirado en la forma en que funciona el sistema nervioso de los animales. Se trata de un sistema de interconexión de neuronas que colaboran entre sí para producir un estímulo de salida.

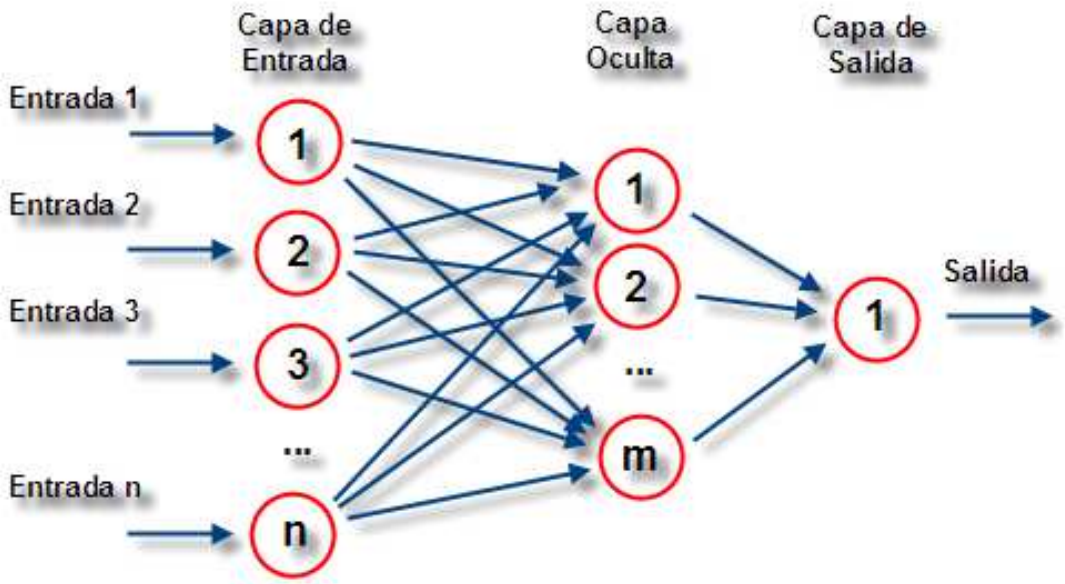


Figura 4 – red neuronal

Fuente [wikipedia2.4.1]

Una de las principales características de las redes neuronales, es que son capaces de trabajar con datos incompletos e incluso paradójicos, que dependiendo del problema puede resultar una ventaja o un inconveniente. Además esta técnica posee dos formas de aprendizaje: supervisado y no supervisado. Las características de las RNA las hacen bastante apropiadas

para aplicaciones en las que no se dispone a priori de un modelo identificable que pueda ser programado, pero se dispone de un conjunto básico de ejemplos de entrada (previamente clasificados o no). Asimismo, son altamente robustas tanto al ruido como a la disfunción de elementos concretos.

Las RNA han sido aplicadas a un número en aumento de problemas en la vida real y de considerable complejidad, donde su mayor ventaja es en la solución de problemas que son bastante complejos para la tecnología actual, tratándose de problemas que no tienen una solución algorítmica o cuya solución algorítmica es demasiado compleja para ser encontrada.

En general, debido a que son parecidas a las del cerebro humano, las RNA son bien nombradas ya que son buenas para resolver problemas que el humano puede resolver pero las computadoras no. Estos problemas incluyen el reconocimiento de patrones y la predicción del tiempo. De cualquier forma, el humano tiene capacidad para el reconocimiento de patrones, pero la capacidad de las redes neuronales no se ve afectada por la fatiga, condiciones de trabajo, estado emocional y compensaciones.

Ejemplo de Modelo Predictivo, fuente [MIT97]

Vamos a predecir si jugaremos a fútbol hoy o no usando la Clasificación ad-hoc árboles de decisión. Hemos recogido los siguientes datos de experiencias pasadas:

Ejemplo	Cielo	Temperatura	Humedad	Viento	¿Jugamos?
1	Soleado	Calor	Alta	Flojo	No
2	Soleado	Calor	Alta	Fuerte	No
3	Nublado	Calor	Alta	Flojo	Sí
4	Lluvia	Suave	Alta	Flojo	Sí
5	Lluvia	Calor	Normal	Flojo	Sí
6	Lluvia	Calor	Normal	Fuerte	No
7	Nublado	Calor	Normal	Fuerte	Sí
8	Soleado	Suave	Alta	Flojo	No
9	Soleado	Calor	Normal	Flojo	Sí
10	Lluvia	Suave	Normal	Flojo	Sí
11	Soleado	Suave	Normal	Fuerte	Sí
12	Nublado	Suave	Alta	Fuerte	Sí
13	Nublado	Calor	Normal	Flojo	Sí
14	Lluvia	Suave	Alta	Fuerte	No

Tabla 1 - Ejemplo de Modelo Predictivo
Fuente [MIT97]

Construimos el árbol de decisión asociado:

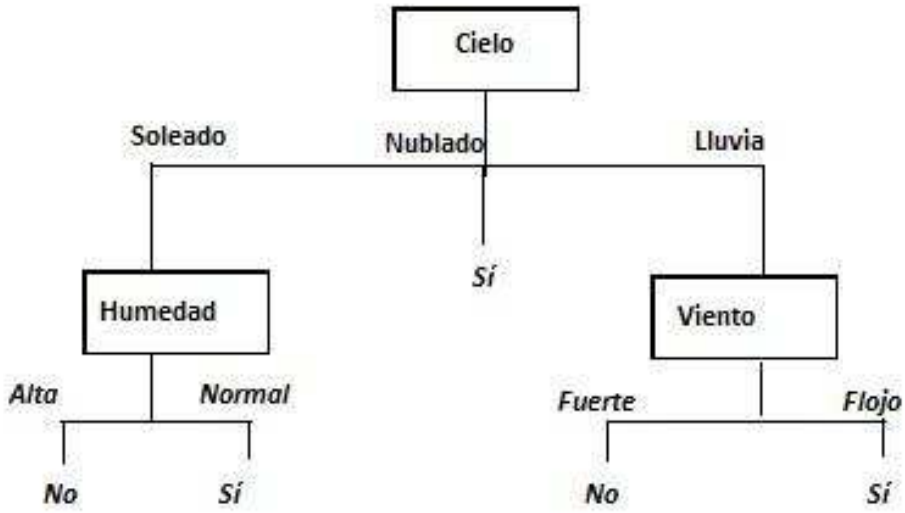


Figura 5 - Ejemplo de Modelo Predictivo
Fuente [MIT97]

Con este modelo, podemos predecir si jugaremos o no, Por ejemplo:

Si Cielo = Soleado Y Temperatura = Calor Y Humedad = Alta Y Viento = Fuerte,

Resultado = No

2.4.2 Modelo descriptivo

Como se comento anteriormente este tipo de modelo trata de Identificar y descubrir patrones que explican y describen los datos. Provee información sobre el estado actual del sistema.

Algunas de las preguntas que se podrían tratar de responder con este tipo de modelo son:

- ¿Los clientes que compran X también compran Y?
- ¿Los niños que no tienen X son muy distintos del resto?
- ¿X e Y son los factores más influyentes en contraer la enfermedad Z?

Los métodos descriptivos se agrupan a grandes rasgos en las siguientes familias:

- Clasificación post-hoc Clustering (Agrupamiento).
- Reglas de Asociación.
- Reducción de la dimensión.
- Análisis exploratorio.
- Escalamiento Multidimensional.

Como en el caso de los modelos predictivos a continuación se exponen las características fundamentales de las familias de métodos enumeradas.

Clasificación post-hoc Clustering (Agrupamiento)

Agrupan datos dentro de un número de clases preestablecidas o no, partiendo de criterios de distancia o similitud, de manera que las clases sean similares entre sí y distintas con las otras clases. Su utilización ha proporcionado significativos resultados en lo que respecta a los clasificadores o reconocedores de patrones, como en el modelado de sistemas. Este método debido a su naturaleza flexible se puede combinar fácilmente con otro tipo de técnica de minería de datos, dando como resultado un sistema híbrido.

Un problema relacionado con el análisis de clúster es la selección de factores en tareas de clasificación, debido a que no todas las variables tienen la misma importancia a la hora de agrupar los objetos. Otro problema de gran importancia y que actualmente despierta un gran interés es la fusión de conocimiento.

Ya que existen múltiples fuentes de información sobre un mismo tema, los cuales no utilizan una categorización homogénea de los objetos. Para poder solucionar estos inconvenientes es necesario fusionar la información a la hora de recopilar, comparar o resumir los datos.

Reglas de Asociación

En minería de datos y aprendizaje automático, las reglas de asociación se utilizan para descubrir hechos que ocurren en común dentro de un determinado conjunto de datos según [MEN03]. Se han investigado ampliamente diversos métodos para aprendizaje de reglas de asociación que han resultado ser muy interesantes para descubrir relaciones entre variables en grandes conjuntos de datos.

Piatetsky-Shapiro [PIA91], describe el análisis y la presentación de reglas 'fuertes' descubiertas en bases de datos utilizando diferentes medidas de interés.

Basado en el concepto de regla fuerte, Agrawal et al [AGR93]. Presentaron un trabajo en el que indicaban las reglas de asociación que descubrían las relaciones entre los datos recopilados a gran escala en los sistemas de terminales de punto de venta de unos supermercados. Por ejemplo, la siguiente regla:

$$\{cebollas, vegetales\} \Rightarrow \{carne\}$$

Encontrada en los datos de ventas de un supermercado, indicaría que un consumidor que compra cebollas y vegetales a la vez, es probable que compre también carne. Esta información se puede utilizar como base para tomar decisiones sobre marketing como precios promocionales para ciertos productos o donde ubicar éstos dentro del supermercado. Además del ejemplo anterior aplicado al análisis de la cesta de la compra. Hoy en día, las reglas de asociación también son de aplicación en otras muchas áreas como el web mining, la detección de intrusos o la bioinformática.

Reducción de la dimensión

Hoy en día es habitual disponer de archivos de elevado tamaño con gran cantidad de variables medidas u observadas en una colección muy amplia de individuos y pretender estudiarlas conjuntamente. Al observar muchas variables sobre muestras de gran tamaño es presumible que una parte de la información recogida pueda ser redundante o que sea excesiva, en cuyo caso los métodos multivariantes de reducción de la dimensión (análisis en

componentes principales, factorial, escalamiento óptimo, etc.) tratan de eliminarla. Estos métodos combinan muchas variables observadas para obtener pocas variables ficticias que las representen con la mínima pérdida de información.

Los métodos de reducción de la dimensión son métodos multivariantes de la interdependencia en el sentido de que todas sus variables tienen una importancia equivalente. Es decir, si ninguna variable destaca como dependiente principal en el objetivo de la investigación. En este caso también deberá tener en cuenta el tipo de variables que se maneja. Si son variables cuantitativas, las técnicas de reducción de la dimensión pueden ser el Análisis de Componentes Principales y el Análisis Factorial. Si son variables cualitativas, puede acudirse al Análisis de Correspondencias y al Escalamiento Óptimo.

Análisis exploratorio

El análisis exploratorio de datos definido por John W. Tukey (E.D.A.: Exploratory data analysis) [TUK83]. Es básicamente el tratamiento estadístico al que se someten las muestras recogidas durante un proceso de investigación en cualquier campo científico. Para mayor rapidez y precisión, todo el proceso suele realizarse por medios informáticos, con aplicaciones específicas para el tratamiento estadístico. Los E.D.A., no necesariamente se llevan a cabo con una base de datos al uso, ni con una hoja de cálculo convencional, no obstante el programa SPSS y R (lenguaje de programación) son las aplicaciones más utilizadas, aunque no las únicas.

Los cálculos estadísticos orientan sobre la fiabilidad de las muestras usadas, aunque no son infalibles, e indican si los resultados obtenidos al calcular las pruebas inferenciales son aceptables, es lo que llamamos nivel de confianza (se debe procurar que éste nunca sea inferior al 95% = 0,95).

Escalamiento Multidimensional

El escalado multidimensional (EMD) (en inglés, multidimensional scaling MDS) se refiere al conjunto de técnicas estadísticas utilizada habitualmente en marketing y ciencias sociales para la visualización y exploración de datos. Es un procedimiento para tomar preferencias y percepciones de los encuestados y representarlos en un diagrama visual. Estos diagramas, llamados mapas perceptuales tienen generalmente dos dimensiones, pero pueden representarse en más de dos. Los consumidores potenciales tienen que comparar pares de productos y hacer juicios sobre sus similitudes.

Mientras otras técnicas (como análisis factorial, análisis discriminativo y análisis conjunto) obtienen dimensiones de las respuestas a los atributos de los productos identificados por el investigador, MDS obtiene las dimensiones de los juicios de los encuestados sobre la similitud de los productos. Esto supone una ventaja importante pues los resultados no dependen de los juicios de los investigadores.

No es necesaria una lista de atributos que debe ser mostrada a los encuestados. Las dimensiones resultantes vienen de los juicios de los encuestados sobre pares de productos. Gracias a estas ventajas, MDS es la técnica más comúnmente utilizada en mapeado perceptual.

Ejemplo de Modelo Descriptivo

En este ejemplo queremos hacer una categorización de un grupo de países basándonos en una serie de atributos, ingreso per cápita, alfabetismo, mortalidad infantil, esperanza de vida**.**

Para ello utilizamos una técnica de Clasificación post-hoc clustering (Agrupamiento). Tenemos los siguientes datos de entrada que se muestran a continuación.

Country	Per capita income	Literacy	Infant mortality	Life expectancy
Brazil	10326	90	23.6	75.4
Germany	39650	99	4.08	79.4
Mozambique	830	38.7	95.9	42.1
Australia	43163	99	4.57	81.2
China	5300	90.9	23	73
Argentina	13308	97.2	13.4	75.3
United Kingdom	34105	99	5.01	79.4
South Africa	10600	82.4	44.8	49.3
Zambia	1000	68	92.7	42.4
Namibia	5249	85	42.3	52.9
Georgia	4200	100	17.36	71
Pakistan	3320	49.9	67.5	65.5
India	2972	61	55	64.7
Turkey	12888	88.7	27.5	71.8
Sweden	34735	99	3.2	80.9
Lithuania	19730	99.6	8.5	73
Greece	36983	96	5.34	79.5
Italy	26760	98.5	5.94	80
Japan	34099	99	3.2	82.6

Tabla 2 - Ejemplo de modelo descriptivo

Fuente [wikibooks2.4.2]

Para este ejemplo, utilizaremos un algoritmo de clustering K-means. Aplicando este algoritmo, el resultado son 3 clústeres que se muestran en la siguiente tabla.

Cluster	Per capita income	Literacy	Infant mortality	Life expectancy	size
1	13370,400	91.58	23,560000	68,96000	5
2	3267,286	70.50	56,251429	58,80000	7
3	35642,143	98.50	4,477143	80,42857	7

Cluster 1	Cluster 2	Cluster 3
Brazil	Mozambique	Germany
South_Africa	China	Australia
Argentina	Namibia	Sweden
Turkey	Georgia	Japan
Lithuania	Pakistan	Greece
	India	Italy
	Zambia	United Kingdom

Tabla 3 – Resultado del ejemplo de modelo descriptivo

Fuente [wikibooks2.4.2]

Por lo tanto, nos da como resultado 3 grupos principales con sus características, estos son:

Clúster 1:

- Formado por 5 Países: Brasil, Sudáfrica, Argentina, Turquía, Lituania.
- Ingresos medios, alto porcentaje de alfabetismo, alta mortalidad infantil, esperanza de vida media.
- Es un grupo formado por países emergentes.

Clúster 2:

- Formado por 7 Países: Mozambique, China, Namibia, Georgia, Pakistán, India, Zambia.
- Ingresos bajos, bajo porcentaje de alfabetismo, alta mortalidad infantil, esperanza de vida baja.
- Es un grupo formado por países subdesarrollado.

Clúster 3:

- Formado por 7 Países: Alemania, Australia, Suecia, Japón, Grecia, Italia, Reino Unido.

- Altos ingresos, alto porcentaje de alfabetismo, baja mortalidad infantil, esperanza de vida alta.
- Es un grupo formado por países desarrollados.

2.5 Interpretación y evaluación de datos

Como viene explicado en el apartado 2.1 la interpretación y evaluación de datos se realiza una vez obtenido el modelo, luego se procede a su validación, donde se comprueba que las conclusiones que arroja son válidas y suficientemente satisfactorias. En el caso de haber obtenido varios modelos mediante el uso de distintas técnicas, se deben comparar los modelos para buscar el que se ajuste mejor al problema. Si ninguno de los modelos alcanza los resultados esperados, debe alterarse alguno de los pasos anteriores para generar nuevos modelos.

En esta fase se evalúa el modelo escogido, no desde el punto de vista general, sino del cumplimiento de los objetivos del negocio. Se debe revisar el proceso teniendo en cuenta los resultados obtenidos. Si el modelo generado es válido en función de los criterios de éxito establecidos en la primera fase y de la precisión del mismo, se procede al despliegue de éste en caso de requerirse. Existen dos planteamientos, uno más interactivo que otro, para extraer información útil con los algoritmos de Minería de Datos:

El primer planteamiento consiste en iniciar un programa, identificar los patrones, normas o funciones y luego hacer que el analista los revise en busca de su valor.

El segundo planteamiento, más interactivo que el anterior, se denomina análisis exploratorio de datos. En él el analista pide que los datos le sean presentados de una forma determinada, los observa, los transforma y los revisa; se mueve hacia delante y hacia detrás, explorando las relaciones que a menudo aparecen, mediante métodos únicos de visualización y por último presenta una respuesta.

Los conocimientos así obtenidos pueden utilizarse posteriormente como entrada para otro análisis y establecer así un ciclo para obtener conclusiones más complejas.

2.6 Data Warehouses y Minería de datos

En el contexto de la informática, un almacén de datos (del inglés data warehouse) es una colección de datos orientada a un determinado ámbito (empresa, organización, etc.), integrado, no volátil y variable en el tiempo. Que ayuda a la toma de decisiones en la entidad en la que se utiliza. Se trata, sobre todo de un expediente completo de una organización. Más allá de la información transaccional y operacional almacenada en una base de datos diseñada para favorecer el análisis y la divulgación eficiente de datos.

Los almacenes de datos (data warehouses) son bases de datos multidimensionales. Son un tipo de tecnología informática que ha tenido un crecimiento de uso constante en las empresas en los últimos años. Esto no es de extrañar, ya que junto al uso de técnicas de minería de datos, las empresas pueden, nada menos, que predecir el futuro, algo vital para el proceso de toma de decisiones.

La mayoría de empresas, tienen sistemas informáticos de bases de datos relacionales para mantener actualizado el trasiego diario de información. Las bases de datos multidimensionales son distintas a las relacionales, ya que guardan información histórica.

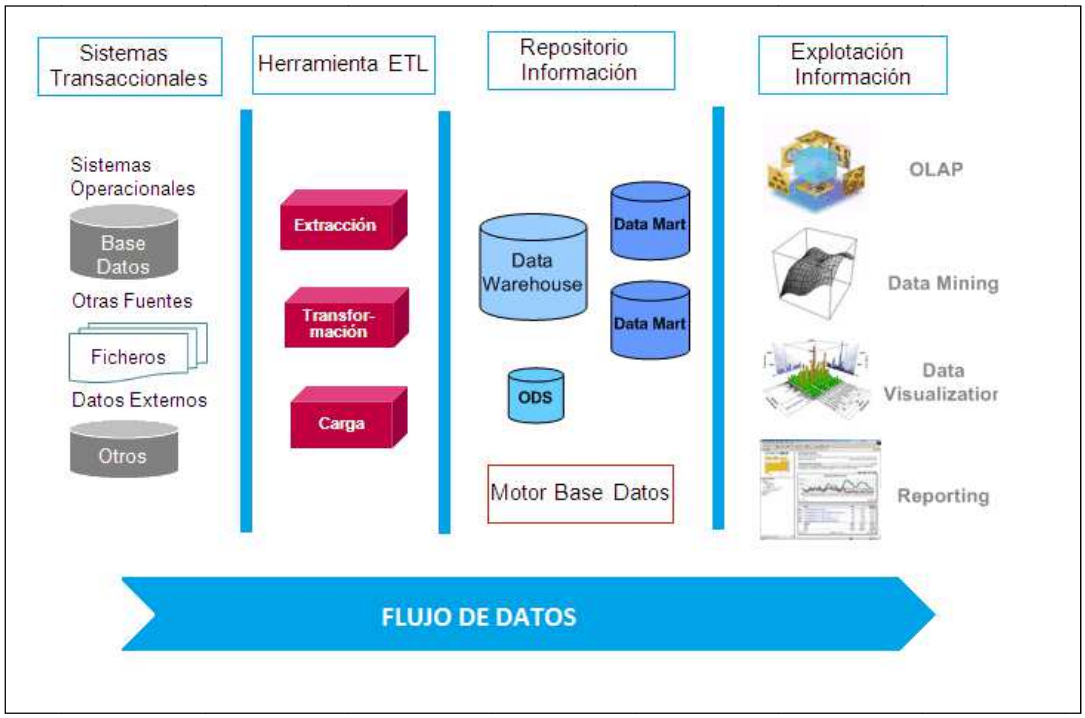


Figura 6 – Data Warehouse

Fuente [dwreview2.6]

Por ejemplo: la compra y devolución de un producto en una tienda dejaría intacta la cantidad de stock del producto en la base de datos relacional, sin embargo, en la base datos multidimensional, tendríamos cómo ha evolucionado el stock a lo largo del tiempo, registrándose la salida y reentrada del producto. Así que los almacenes de datos manejan información histórica y las bases de datos relacionales permiten que la empresa pueda hacer su trabajo diario.

El término data warehouse fue acuñado por primera vez por Bill Inmon [DYC00], y se traduce literalmente como almacén de datos. No obstante, y como cabe suponer, es mucho más que eso. Según definió el propio *Bill Inmon*, un data warehouse se caracteriza por ser:

- **Integrado:** los datos almacenados en el data warehouse deben integrarse en una estructura consistente, por lo que las inconsistencias existentes entre los diversos sistemas operacionales deben ser eliminadas. La información suele estructurarse también en distintos niveles de detalle para adecuarse a las distintas necesidades de los usuarios.
- **Temático:** sólo los datos necesarios para el proceso de generación del conocimiento del negocio se integran desde el entorno operacional. Los datos se organizan por temas para facilitar su acceso y entendimiento por parte de los usuarios finales. Por ejemplo, todos los datos sobre clientes pueden ser consolidados en una única tabla del data warehouse. De esta forma, las peticiones de información sobre clientes serán más fáciles de responder dado que toda la información reside en el mismo lugar.
- **Histórico:** el tiempo es parte implícita de la información contenida en un data warehouse. En los sistemas operacionales, los datos siempre reflejan el estado de la actividad del negocio en el momento presente. Por el contrario, la información almacenada en el data warehouse sirve, entre otras cosas, para realizar análisis de tendencias. Por lo tanto, el data warehouse se carga con los distintos valores que toma una variable en el tiempo para permitir comparaciones.
- **No volátil:** el almacén de información de un data warehouse existe para ser leído, pero no modificado. La información es por tanto permanente, significando la actualización del data warehouse la incorporación de los últimos valores que tomaron las distintas variables contenidas en él sin ningún tipo de acción sobre lo que ya existía.

Como podemos observar en la figura 6, la Minería de Datos puede obtener los datos de un Data Warehouse, Sin embargo no es necesario construir un Data Warehouse para hacer minería de datos. Ya que también puede aplicarse minería de datos directamente a las bases de datos.

Sin embargo, la diferencia radica en la forma en cómo un data warehouse estructura los datos de tal manera que facilita la aplicación de la minería de datos, por lo que en muchos casos es muy deseable tener un almacén del datos para llevar a cabo la minería de datos.

Esencialmente, un warehouse organiza y estructura los datos eficazmente para realizar técnicas de minería de datos sobre ellos. Hay que tener en cuenta que es muy interesante tener un Data Warehouse, pero esto no significa que sea imprescindible.

Además, los data warehouse disponen de capacidades para el apoyo a la toma de decisiones. Algunos data warehouse llevan a cabo predicciones y tendencias. En este caso los data warehouse llevan a cabo algunas de las funciones de minería de datos.

2.6.1 Procesos ETL

Los almacenes de datos suelen necesitar varios cientos de GB para almacenar toda esa información. Su diseño es muy diferente al de las bases de datos relacionales, ya que con este cambio de diseño ahorramos mucho tiempo a la hora de consultar todos esos GB de datos. La información la obtienen de las bases de datos relacionales, utilizando los llamados procesos ETL Extracción, transformación y carga:

- Extracción: este es el primer paso de obtención de información de las distintas fuentes tanto internas como externas.
- Transformación: una vez que la información es extraída hacia el área de tráfico de datos, hay posibles paso de transformación como; filtrado, limpieza, depuración, homogeneización y agrupación de la información, seleccionar únicamente los campos necesarios para el Data Warehouse, combinar fuentes de datos.
- Carga: al final del proceso de transformación, organización y actualización de los datos y los metadatos en la base de datos.

2.6.2 DATA MARTS

Los usuarios a menudo realizaban amplias operaciones de informes y análisis de datos sobre un subconjunto relativamente pequeño de todo el Data Warehouse. Asimismo, era muy probable que los usuarios repitieran las mismas operaciones sobre el mismo subconjunto de datos cada vez que era actualizado. Además, algunas de esas actividades involucraban la creación de nuevos esquemas y datos con actualizaciones posteriores a esos nuevos datos.

La ejecución repetida de tales operaciones sobre el mismo subconjunto de todo el almacén no era muy eficiente; por lo tanto, pareció buena idea construir algún tipo de "almacén" limitado de propósito general que estuviera hecho a la medida de ese propósito. Además, en algunos casos sería posible extraer y preparar los datos requeridos directamente a partir de las fuentes locales, lo que proporcionaba un acceso más rápido a los datos que si tuvieran que ser sincronizados con los demás datos cargados en todo el Data Warehouse. Dichas consideraciones condujeron al concepto de Data Marts.

Se puede definir como "un almacén de datos especializado, orientado a un tema, integrado, volátil y variante en el tiempo para apoyar un subconjunto específico de decisiones de administración". La principal diferencia entre un Data Mart y un Data Warehouse es que el Data Mart es especializado y volátil. Especializado quiere decir que contiene datos para dar apoyo (solamente) a un área específica de análisis de negocios; por volátil se entiende que los usuarios pueden actualizar los datos e incluso, posiblemente, crear nuevos datos (es decir, nuevas tablas) para algún propósito. En síntesis, se puede decir que los data marts son pequeños data warehouse centrados en un tema o un área de negocio específico dentro de una organización.

2.6.3 Metadatos

Otra característica del data warehouse es que contiene metadatos, es decir, datos sobre los datos. Los metadatos documenta, entre otras cosas la procedencia de la información, su periodicidad de refresco, su fiabilidad, forma de cálculo, qué tablas existen en una base de datos, qué columnas posee cada una de las tablas y qué tipo de datos se pueden almacenar.

Los datos son de interés para el usuario final, el metadato es de interés para los programas que tienen que manejar estos datos. Sin embargo, el rol que cumple el metadato en un entorno de almacén de datos es muy diferente al rol que cumple en los ambientes operacionales.

En el ámbito de los data warehouse el metadato juega un papel fundamental, permiten simplificar y automatizar la obtención de la información desde los sistemas operacionales a los sistemas informacionales, su función consiste en recoger todas las definiciones de la organización y el concepto de los datos en el almacén de datos, debe contener toda la información concerniente a:

- Tablas.
- Columnas de tablas.
- Relaciones entre tablas.
- Jerarquías y Dimensiones de datos.
- Entidades y Relaciones.

2.6.4 Ventajas e inconvenientes de los almacenes de datos

Hay muchas ventajas por las que es recomendable usar un almacén de datos. Algunas de ellas son:

- Los almacenes de datos facilitan en gran medida el acceso a una gran variedad de datos a los usuarios finales
- Facilitan el funcionamiento de las aplicaciones de los sistemas de apoyo a la decisión tales como informes de tendencia, informes de excepción, informes que muestran los resultados reales frente a los objetivos planteados a priori.
- Los almacenes de datos pueden trabajar en conjunto y, por lo tanto, aumentar el valor operacional de las aplicaciones empresariales, en especial la gestión de relaciones con clientes.

Utilizar almacenes de datos también plantea algunos inconvenientes, algunos de ellos son:

- Implementar un Data warehouse implica un alto coste y no suelen ser estáticos sino que a lo largo de su vida los almacenes de datos pueden suponer altos costos de mantenimiento.
- Los almacenes de datos pueden quedar obsoletos en cualquier momento.
- A menudo existe una delgada línea entre los almacenes de datos y los sistemas operacionales. Hay que determinar qué funcionalidades de estos se pueden aprovechar y cuáles se deben implementar en el data warehouse, resultaría costoso implementar operaciones no necesarias o dejar de implementar alguna que sí vaya a necesitarse.

2.7 Relación con otras disciplinas

Esencialmente la minería de datos consiste en hacer una serie de consultas, cuyo resultado puede depender, a su vez, de de la respuesta de anteriores consultas. La Minería de datos, con una fuerte base matemática (sobre todo en modelización) e informática, involucra a muchas otras disciplinas tales como:

- Bases de Datos es la esencia de la minería de datos, los datos que tenemos que tratar y/o analizar se guardan como estructuras y así poder tener acceso a dichos datos de forma fácil y rápida además contribuye con las técnicas deductivas de procesamiento de consultas.
- Inteligencia Artificial comprende el Aprendizaje automático, lógica difusa, programación lógica, redes neuronales, algoritmos genéticos, etc.

- Estadística La estadística juega un importante papel en el análisis de los datos, e incluso también en el aprendizaje automático. Debido a esto, no se puede estudiar la minería de Datos sin un buen conocimiento de la estadística, la estadística nos ofrece un gran número de herramientas que se usan en la minería de datos como: regresiones, probabilidades, razonamiento probabilístico, análisis clúster, etc.
- Investigación Operativa comprende modelado, algoritmos y toma de decisiones.
- Análisis matemático por medio de técnicas matemáticas tales como las Series Temporales y visualización para lograr una minería de datos interactiva.
- Visualización de datos: son tecnologías que nos permiten mostrar gráficamente los datos almacenados en las bases de datos, los datos en las bases de datos son unos conjuntos de filas y columnas o estructuras de datos son valores numéricos, Los modelos de visualización pueden ser bidimensionales, tridimensionales o incluso multidimensionales., Las herramientas de visualización toman estos datos y trazan con ellos algún tipo de gráfico. Así, las herramientas de visualización ayudan de forma interactiva a la Minería de Datos.
- Herramientas apoyo a la toma de decisión comprende herramientas de evaluación del rendimiento, planificación, organización, árboles de decisión, etc.
- Heurística Algoritmos genéticos, métodos del vecino más cercano, etc.
- Computación paralela / distribuida cómputo de alto desempeño, mejora de desempeño de algoritmos debido a su complejidad y a la cantidad de datos, para mejorar el rendimiento de los algoritmos de minería de datos, y para conseguir mejorar el rendimiento utilizamos arquitecturas informáticas desde sistemas con un único procesador hasta sistemas multiprocesador. Los sistemas de multiprocesamiento pueden estar formados por sistemas distribuidos o por sistemas centralizados de multiprocesadores con memoria compartida, o con multiprocesadores sin memoria compartida.
- Arquitectura de ordenadores para escalar las técnicas de Minería de Datos se necesita hardware y software apropiado, una buena arquitectura nos permite ofrecer un entorno adecuado para la aplicación de las técnicas de Minería de Datos.

2.8 Extensiones de minería de Datos

2.8.1 Web Mining

Una de las extensiones del data mining consiste en aplicar sus técnicas a documentos y servicios del Web, lo que se llama web mining (minería de web). Todos los que visitan un sitio en Internet dejan huellas digitales (direcciones de IP, navegador, galletas, etc.) que los servidores automáticamente almacenan en una histórico de accesos (log). Las herramientas de web mining analizan y procesan estos logs para producir información significativa, por ejemplo, cómo es la navegación de un cliente antes de hacer una compra en línea.

Debido a que los contenidos de Internet consisten en varios tipos de datos, como texto, imagen, vídeo, metadatos o hiperligas, investigaciones recientes usan el término multimedia data mining(minería de datos multimedia) como una instancia del web mining para tratar ese tipo de datos. Los accesos totales por dominio, horarios de accesos más frecuentes y visitas por día, entre otros datos, son registrados por herramientas estadísticas que complementan todo el proceso de análisis del web mining.

2.8.2 Text Mining

Este campo de estudio es muy vasto, por lo que técnicas como la categorización de texto, el procesamiento de lenguaje natural, la extracción y recuperación de la información o el aprendizaje automático, entre otras, apoyan al text mining (minería de texto).

El text mining se refiere a examinar una colección de documentos y descubrir información no contenida en ningún documento individual de la colección; en otras palabras, trata de obtener información sin haber partido de algo.

2.8.3 Minería de datos aplicada a las redes sociales

Las redes sociales son estructuras sociales compuestas de grupos de personas, las cuales están conectadas por uno o varios tipos de relaciones, tales como amistad, intereses comunes o que comparten conocimientos e ideas. Puede haber muchos tipos de lazos entre los nodos. Las redes sociales operan en muchos niveles, desde las relaciones de amistad hasta las relaciones de organizaciones a nivel estatal (se habla en este caso de Redes políticas).

La minería de datos aplicada a las redes sociales pretende incrementar la satisfacción de la experiencia de los usuarios, que reciben ofertas de mayor interés al haber sido diseñadas previamente en función de los datos obtenidos, los usuarios pueden contactarse con personas más similares a ellos, les llegan promociones más acordes a sus gustos, necesidades y posibilidades y mejoras del servicio en varios aspectos.

2.9 Aplicaciones De La Minería De Datos

2.9.1 Detección de fraudes

Esta es una aplicación que puede ser considerada como una técnica de clasificación. En efecto, cuando el algoritmo analiza una gran cantidad de transacciones, el mismo tratará de categorizar aquellas que sean ilegítimas mediante la identificación de ciertas características que estas últimas tengan en común. Esto puede ser usado en las corporaciones para prevenir que se culmine un proceso que muestre pertenecer a una "clase" peligrosa.

2.9.2 Análisis de riesgos en créditos

Esta es una aplicación similar a la anterior, pero con la ventaja de de la existencia de maneras tradicionales para realizarlo. El clásico procedimiento de asignación de puntos puede ser complementado y mejorado con la ayuda de la minería de datos.

2.9.3 Investigaciones Espaciales

Debido a la gran contribución a estas tareas por parte del reconocimiento de imágenes y los PRE-procesamientos involucrados, esta aplicación también puede considerarse como perteneciente al área del reconocimiento de patrones de imágenes (Pattern Recognition).

Proyecto SKYCAT [SKYCAT2.9.3]

Durante seis años, el Second Palomar Observatory Sky Survey (POSS-II) coleccionó tres terabytes de imágenes que contenían aproximadamente dos millones de objetos en el cielo. Tres mil fotografías fueron digitalizadas a una resolución de 16 bits por píxel con 23.040 x 23.040 píxeles por imagen.

El objetivo era formar un catálogo de todos esos objetos. El sistema Sky Image Cataloguing and Analysis Tool (SKYCAT) se basa en técnicas de agrupación (clustering) y árboles de decisión para poder clasificar los objetos en estrellas, planetas, sistemas, galaxias, etc. Con una alta confiabilidad

(Fayyad y otros, 1996). Los resultados han ayudado a los astrónomos a descubrir dieciséis nuevos quásars con corrimiento hacia el rojo que los incluye entre los objetos más lejanos del universo y, por consiguiente, más antiguos. Estos quásars son difíciles de encontrar y permiten saber más acerca de los orígenes del universo.

2.9.4 Minería de texto

Con billones de páginas en la red, se requieren de nuevas tecnologías para encontrar, clasificar y detectar particulares patrones en la información disponible. La esencia de los métodos de la minería de datos aplicados a los datos numéricos, puede también ser aplicada a datos de texto.

2.9.5 Negocios

La minería de datos puede contribuir significativamente en las aplicaciones de administración empresarial basada en la relación con el cliente. En lugar de contactar con el cliente de forma indiscriminada a través de un centro de llamadas o enviando cartas, sólo se contactará con aquellos que se perciba que tienen una mayor probabilidad de responder positivamente a una determinada oferta o promoción, Por lo general, las empresas que emplean minería de datos ven rápidamente el retorno de la inversión, pero también reconocen que el número de modelos predictivos desarrollados puede crecer muy rápidamente.

En lugar de crear modelos para predecir qué clientes pueden cambiar, la empresa podría construir modelos separados para cada región y/o para cada tipo de cliente. También puede querer determinar que clientes van a ser rentables durante una ventana de tiempo (una quincena, un mes,...) y sólo enviar las ofertas a las personas que es probable que sean rentables. Para mantener esta cantidad de modelos, es necesario gestionar las versiones de cada modelo y pasar a una minería de datos lo más automatizada posible.

2.9.6 En los Clubes Deportivos

Los equipos de la NBA utilizan aplicaciones inteligentes para apoyar a su cuerpo de entrenadores. El Advanced Scout es un software que emplea técnicas de data mining y que han desarrollado investigadores de IBM para detectar patrones estadísticos y eventos raros. Tiene una interfaz gráfica muy amigable orientada a un objetivo muy específico: analizar el juego de los equipos de la National Basketball Association (NBA).

El software utiliza todos los registros guardados de cada evento en cada juego: pases, encestes, rebotes y doble marcaje (double team) a un jugador por el equipo contrario, entre otros. El objetivo es ayudar a los entrenadores a aislar eventos que no detectan cuando observan el juego en vivo o en película.

Un resultado interesante fue uno hasta entonces no observado por los entrenadores de los Knicks de Nueva York. El doble marcaje a un jugador puede generalmente dar la oportunidad a otro jugador de encestar más fácilmente. Sin embargo, cuando los Bulls de Chicago jugaban contra los Knicks, se encontró que el porcentaje de encestes después de que al centro de los Knicks, Patrick Ewing, le hicieran doble marcaje era extremadamente bajo, indicando que los Knicks no reaccionaban correctamente a los dobles marcajes.

Para saber el porqué, el cuerpo de entrenadores estudió cuidadosamente todas las películas de juegos contra Chicago. Observaron que los jugadores de Chicago rompían su doble marcaje muy rápido de tal forma que podían tapar al encestador libre de los Knicks antes de prepararse para efectuar su tiro. Con este conocimiento, los entrenadores crearon estrategias alternativas para tratar con el doble marcaje.

IBM ofreció el Advanced Scout a la NBA, que se convirtió así en un patrocinador corporativo. La NBA dio a sus veintinueve equipos la oportunidad de aplicarlo. Dieciocho equipos lo están haciendo hasta el momento obteniendo descubrimientos interesantes.

2.9.7 Hábitos de compra en supermercados

El ejemplo clásico de aplicación de la minería de datos tiene que ver con la detección de hábitos de compra en supermercados. Un estudio muy citado detectó que los viernes había una cantidad inusualmente elevada de clientes que adquirían a la vez pañales y cerveza. Se detectó que se debía a que dicho día solían acudir al supermercado padres jóvenes cuya perspectiva para el fin de semana consistía en quedarse en casa cuidando de su hijo y viendo la televisión con una cerveza en la mano. El supermercado pudo incrementar sus ventas de cerveza colocándolas próximas a los pañales para fomentar las ventas compulsivas.

2.9.8 Patrones de fuga

Un ejemplo más habitual es el de la detección de patrones de fuga. En muchas industrias —como la banca, las telecomunicaciones, etc. — existe un comprensible interés en detectar cuanto antes aquellos clientes que puedan estar pensando en rescindir sus contratos para, posiblemente, pasarse

a la competencia. A estos clientes —y en función de su valor— se les podrían hacer ofertas personalizadas, ofrecer promociones especiales, etc., con el objetivo último de retenerlos. La minería de datos ayuda a determinar qué clientes son los más proclives a darse de baja estudiando sus patrones de comportamiento y comparándolos con muestras de clientes que, efectivamente, se dieron de baja en el pasado.

2.9.9 Fraudes

Un caso análogo es el de la detección de transacciones de blanqueo de dinero o de fraude en el uso de tarjetas de crédito o de servicios de telefonía móvil e, incluso, en la relación de los contribuyentes con el fisco. Generalmente, estas operaciones fraudulentas o ilegales suelen seguir patrones característicos que permiten, con cierto grado de probabilidad, distinguirlas de las legítimas y desarrollar así mecanismos para tomar medidas rápidas frente a ellas.

2.9.10 Prediciendo el tamaño de las audiencias televisivas.

La British Broadcasting Corporation (BBC) del Reino Unido emplea un sistema para predecir el tamaño de las audiencias televisivas para un programa propuesto, así como el tiempo óptimo de exhibición.

El sistema utiliza redes neuronales y árboles de decisión aplicados a datos históricos de la cadena para determinar los criterios que participan según el programa que hay que presentar. La versión final se desempeña tan bien como un experto humano con la ventaja de que se adapta más fácilmente a los cambios porque es constantemente reentrenada con datos actuales.

2.9.11 Recursos humanos

La minería de datos también puede ser útil para los departamentos de recursos humanos en la identificación de las características de sus empleados de mayor éxito. La información obtenida puede ayudar a la contratación de personal, centrándose en los esfuerzos de sus empleados y los resultados obtenidos por éstos. Además, la ayuda ofrecida por las aplicaciones para Dirección estratégica en una empresa se traducen en la obtención de ventajas a nivel corporativo, tales como mejorar el margen de beneficios o compartir objetivos; y en la mejora de las decisiones operativas, tales como desarrollo de planes de producción o gestión de mano de obra.

2.9.12 Comportamiento en Internet

También es un área en boga el del análisis del comportamiento de los visitantes —sobre todo, cuando son clientes potenciales— en una página de Internet. O la utilización de la información —obtenida por medios más o menos legítimos— sobre ellos para ofrecerles propaganda adaptada específicamente a su perfil. O para, una vez que adquieren un determinado producto, saber inmediatamente qué otro ofrecerle teniendo en cuenta la información histórica disponible acerca de los clientes que han comprado el primero.

2.9.13 Terrorismo

La minería de datos ha sido citada como el método por el cual la unidad Able Danger del Ejército de los EE.UU. había identificado al líder de los atentados del 11 de septiembre de 2001, Mohammed Atta, y a otros tres secuestradores del "11-S" como posibles miembros de una célula de Al Qaeda que operan en los EE.UU. más de un año antes del ataque. Se ha sugerido que tanto la Agencia Central de Inteligencia y sus homóloga canadiense, Servicio de Inteligencia y Seguridad Canadiense, también han empleado este método.

2.9.14 Juegos

Desde comienzos de la década de 1960, con la disponibilidad de oráculos para determinados juegos combinacionales, también llamados finales de juego de tablero (por ejemplo, para las tres en raya o en finales de ajedrez) con cualquier configuración de inicio, se ha abierto una nueva área en la minería de datos que consiste en la extracción de estrategias utilizadas por personas para estos oráculos. Los planteamientos actuales sobre reconocimiento de patrones, no parecen poder aplicarse con éxito al funcionamiento de estos oráculos. En su lugar, la producción de patrones perspicaces se basa en una amplia experimentación con bases de datos sobre esos finales de juego, combinado con un estudio intensivo de los propios finales de juego en problemas bien diseñados y con conocimiento de la técnica (datos previos sobre el final del juego). Ejemplos notables de investigadores que trabajan en este campo son Berlekamp en el juego de puntos-y-cajas (o Timbiriche) y John Nunn en finales de ajedrez.

2.9.15 Ciencia e Ingeniería

En los últimos años la minería de datos se está utilizando ampliamente en diversas áreas relacionadas con la ciencia y la ingeniería. Algunos ejemplos de aplicación en estos campos son:

- **Genética:** En el estudio de la genética humana, el objetivo principal es entender la relación cartografía entre las partes y la variación individual en las secuencias del ADN humano y la variabilidad en la susceptibilidad a las enfermedades. En términos más llanos, se trata de saber cómo los cambios en la secuencia de ADN de un individuo afectan al riesgo de desarrollar enfermedades comunes (como por ejemplo el cáncer). Esto es muy importante para ayudar a mejorar el diagnóstico, prevención y tratamiento de las enfermedades. La técnica de minería de datos que se utiliza para realizar esta tarea se conoce como "reducción de dimensionalidad multifactorial" [XIN07].

- **Ingeniería eléctrica:** En el ámbito de la ingeniería eléctrica, las técnicas minería de datos han sido ampliamente utilizadas para monitorizar las condiciones de las instalaciones de alta tensión. La finalidad de esta monitorización es obtener información valiosa sobre el estado del aislamiento de los equipos. Para la vigilancia de las vibraciones o el análisis de los cambios de carga en transformadores se utilizan ciertas técnicas para agrupación de datos (clustering) tales como los Mapas Auto-Organizativos (SOM, Self-organizing map) [KOH07] [KOH82]. Estos mapas sirven para detectar condiciones anormales y para estimar la naturaleza de dichas anomalías.

Capítulo 3

3. Aprendizaje automático

El aprendizaje automático o aprendizaje de máquinas es una rama de la Inteligencia artificial cuyo objetivo es desarrollar técnicas que permitan a las computadoras aprender. De forma más concreta, se trata de crear programas capaces de generalizar comportamientos a partir de una información no estructurada suministrada en forma de ejemplos.

Es, por lo tanto, un proceso de inducción del conocimiento. En muchas ocasiones el campo de actuación del aprendizaje automático se solapa con el de la estadística, ya que las dos disciplinas se basan en el análisis de datos. Sin embargo, el aprendizaje automático se centra más en el estudio de la complejidad computacional de los problemas.

Es un concepto primordial y diferenciador de las técnicas estadísticas más clásicas, que fue concebido hace aproximadamente cuatro décadas con el objetivo de desarrollar métodos computacionales que implementarían varias formas de aprendizaje. En particular, mecanismos capaces de inducir conocimiento a partir de datos. Ya que el desarrollo de software ha llegado a ser

uno de los principales cuellos de botella de la tecnología informática de hoy en día, la idea de introducir conocimiento por medio de ejemplos parece particularmente atractiva al sentido común.

Tal forma de inducción de conocimiento es deseable en problemas que carecen de solución algorítmica eficiente, son vagamente definidos, o informalmente especificados. Ejemplos de tales problemas pueden ser la diagnosis médica, el reconocimiento de patrones visuales o la detección de regularidades en enormes cantidades de datos.

Los algoritmos de aprendizaje automático pueden clasificarse en dos grandes categorías: métodos de caja negra (o sin modelo), tales como redes neuronales o los métodos bayesianos, y métodos orientados al conocimiento, tales como los que generan árboles de decisión, reglas de asociación, o reglas de decisión. La propuesta de caja negra desarrolla su propia representación del conocimiento, que no es visible desde el exterior.

Los métodos orientados al conocimiento, por el contrario, construyen una estructura simbólica del conocimiento que intenta ser útil desde el punto de vista de la funcionalidad, pero también descriptiva desde la perspectiva de la inteligibilidad. Existen también métodos para extraer reglas comprensibles a partir de estas cajas negras, con lo que en realidad ambas categorías pueden ser útiles para la extracción de conocimiento.

Lógicamente, las áreas del aprendizaje automático y la minería de datos se solapan en gran medida, en cuanto a los problemas que tratan y a los algoritmos que utilizan. No obstante, la minería de datos tiene un mayor enfoque en el conocimiento comprensible a partir de grandes cantidades de información, mientras que el aprendizaje automático se orienta más a la tarea del aprendizaje propiamente dicho, buscando en algunos casos estrategias o heurísticas, más que el propio conocimiento comprensible.

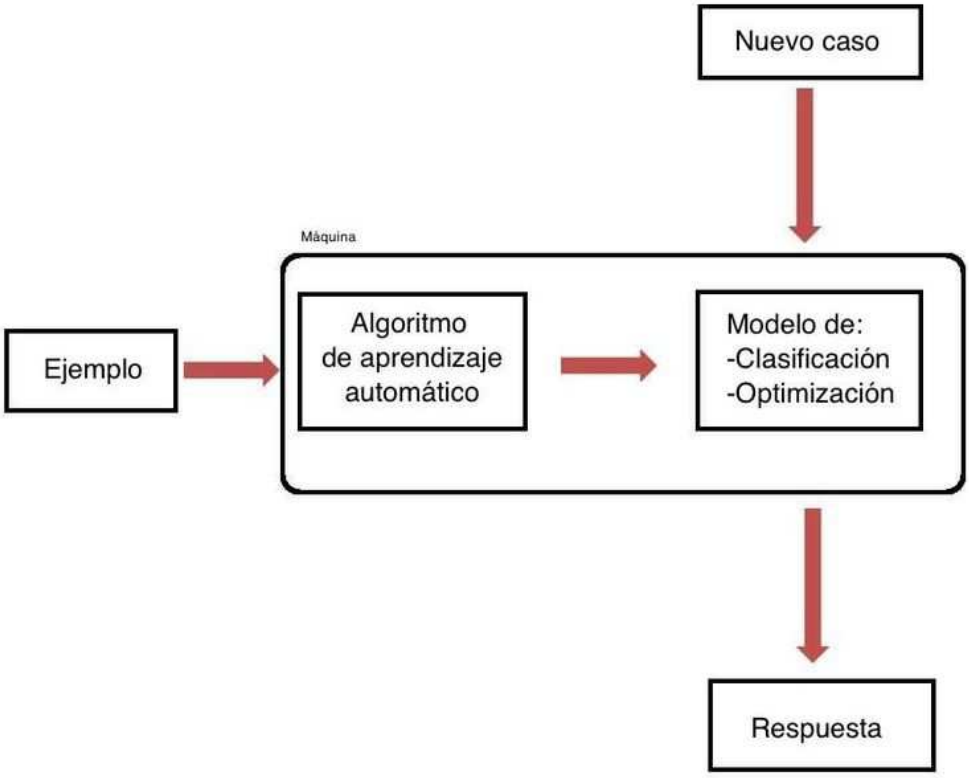


Figura 7 –Diagrama aprendizaje automático

Fuente [wikipedia3]

El aprendizaje automático puede ser visto como un intento de automatizar algunas partes del método científico mediante métodos matemáticos.

El aprendizaje automático tiene una amplia gama de aplicaciones, incluyendo motores de búsqueda, diagnósticos médicos, detección de fraude en el uso de tarjetas de crédito, análisis del mercado de valores, clasificación de secuencias de ADN, reconocimiento del habla y del lenguaje escrito, juegos y robótica.

3.1 Tipos de aprendizaje automático

1. **Aprendizaje supervisado**

   El algoritmo produce una función que establece una correspondencia entre las entradas y las salidas deseadas del sistema. Un ejemplo de este tipo de algoritmo es el problema de clasificación, donde el sistema de aprendizaje trata de etiquetar (clasificar) una serie de vectores utilizando una entre varias categorías (clases). La base de conocimiento del sistema está formada por ejemplos de etiquetados anteriores. Este tipo de aprendizaje puede llegar a ser muy útil en problemas de investigación biológica, biología computacional y bioinformática.

2. **Aprendizaje no supervisado**

   Todo el proceso de modelado se lleva a cabo sobre un conjunto de ejemplos formado tan sólo por entradas al sistema. No se tiene información sobre las categorías de esos ejemplos. Por lo tanto, en este caso, el sistema tiene que ser capaz de reconocer patrones para poder etiquetar las nuevas entradas.

3. **Aprendizaje semisupervisado**

   Este tipo de algoritmos combinan los dos algoritmos anteriores para poder clasificar de manera adecuada. Se tiene en cuenta los datos marcados y los no marcados.

4. **Aprendizaje por refuerzo**

   El algoritmo aprende observando el mundo que le rodea. Su información de entrada es el feedback o retroalimentación que obtiene del mundo exterior como respuesta a sus acciones. Por lo tanto, el sistema aprende a base de ensayo-error.

5. **Transducción**

   Similar al aprendizaje supervisado, pero no construye de forma explícita una función. Trata de predecir las categorías de los futuros ejemplos basándose en los ejemplos de entrada, sus respectivas categorías y los ejemplos nuevos al sistema.

6. **Aprendizaje multitarea**

   Métodos de aprendizaje que usan conocimiento previamente aprendido por el sistema de cara a enfrentarse a problemas parecidos a los ya vistos.

3.2 Algoritmos de aprendizaje automático

Hay infinidad de algoritmos que reúnen diversas técnicas de aprendizaje automático, a continuación se muestra una clasificación de los algoritmos más representativos. Dichos algoritmos se dividen en dos grandes grupos:

1. **Algoritmos aprendizaje supervisado**

   - Árboles de Decisión: ID3 [QUI86], C4.5 [QUI93], CART [BRE84]
   - Aprendizaje basado en instancias: k-NN [HOD89]
   - Redes de Neuronas Artificiales (ANN): Perceptron [ROS57] , backpropagation [BRY63]
   - Clasificadores Estadísticos: Naive Bayes [BAY63]
   - Basados en kernels: SVM [VAP95]

2. **Algoritmos aprendizaje no supervisado**

   - Clustering : K-MEANS [MAC67] [STE57] , EM [DEM77], DBSCAN [EST96]
   - Clustering conceptual jerárquico : Cobweb [FIS87] [FIS87b]
   - Redes de Neuronas Artificiales (ANN): Mapas auto-organizados (SOM) [KOH07] [KOH82]

En este capítulo vamos a estudiar un conjunto de dichos algoritmos, tanto de aprendizaje supervisado como no supervisado

Algoritmos de aprendizaje supervisado:

1. **ID3:** Este algoritmo genera clasificadores expresados en términos de árboles de decisión.
2. **Bayes** ingenuo (*Naive Bayes*): Dado un conjunto de objetos, que pertenecen a una clase conocida, construye una regla que permite asignar objetos futuros a una clase.
3. **SVM** (*Support Vector Machine*): Mediante el aprendizaje este algoritmo trata de encontrar la mejor función de clasificación para distinguir en miembros de distintas clases.

Algoritmos de aprendizaje no supervisado:

1. **El algoritmo de *k*-medias:** Es un método simple iterativo que particiona un conjunto de datos en un número pre-especificado de conglomerados.
2. **El algoritmo EM:** Es utilizado para clasificar datos de naturaleza continua y para estimar su correspondiente función de densidad.

3.3 Algoritmo ID3

El algoritmo ID3 "Iterative Dichotomizer (version) 3" (J.R. Quinlan, 1979) [QUI86]. Toma objetos de una clase conocida y los describe en términos de una colección fija de propiedades o de atributos, y produce un árbol de decisión sobre estos atributos que clasifica correctamente todos los objetos.

Es el algoritmo de aprendizaje simbólico que ha despertado mayor interés dada su utilidad en los problemas de clasificación del mundo real, Este pertenece a la familia de algoritmos de TDIDT (Top Down Inductive of Decision Trees) según [QUI86]. El cual genera un árbol de decisión a partir de un conjunto de experiencias.

Es uno de los algoritmos de aprendizaje automático más conocidos, basado en "ejemplos". Trabaja con datos simbólicos, en contraposición a los datos numéricos, y se basa en la obtención de un árbol de decisión ,a partir del cual se obtienen una serie de reglas de producción, capaces de representar un dominio o universo determinado, generando conocimiento independiente de dicho dominio (el sistema de aprendizaje parte de un estado inicial del dominio escogido en el que no existe conocimiento de partida, extrayendo patrones comunes de entre los ejemplos utilizados, a partir de los cuales genera una base de conocimientos de aplicación a dicho dominio).

El algoritmo ID3 genera lo que se conoce como reglas que solo atienden a dos posibles estados (verdadero-falso, positivo-negativo, 0-1, etc.), y que tienen por tanto un carácter bivalente, a diferencia de las reglas "borrosas", que permiten representar un rango infinito de valores entre dos extremos de una escala, como las que se obtienen mediante algoritmos ID3 "extendidos" (ID4, ID5, ID5R, C4.5, C5).

El ID3 fue presentado como descendiente del CLS (creado por Hunt) [HUN66]. El ID3, como contrapartida de su antecesor es un mecanismo mucho más simple para el descubrimiento de una colección de objetos pertenecientes a dos o más clases. Cada objeto debe estar descripto en términos de un conjunto fijo de atributos, cada uno de los cuales cuenta con su conjunto de posibles valores de atributos. Por ejemplo, el atributo humedad puede tener los valores {alta, baja}, y el atributo clima, {soleado, nublado, lluvioso}.

3.3.1 Pseudocódigo del ID3

ID3 (Ejemplos, Atributo-objetivo, Atributos)

Si *todos los ejemplos son positivos* devolver un nodo positivo.

Si *todos los ejemplos son negativos* devolver un nodo negativo.

Si *Atributos está vacío* devolver el voto mayoritario del valor del atributo objetivo en Ejemplos.

En otro caso

Sea A Atributo el MEJOR de atributos

Para cada v valor del atributo hacer

Sea Ejemplos (v) el subconjunto de ejemplos cuyo valor de atributo A es v

Si Ejemplos (v) está vacío devolver un nodo con el voto mayoritario del Atributo objetivo de Ejemplos

Sino Devolver ID3 (Ejemplos (v), Atributo- objetivo, Atributos/{A})

Según fuente [QUI86]

La construcción del árbol se hace forma recursiva, siendo las tres primeras líneas y la penúltima los casos base que construyen los nodos hojas.

3.3.2. Medidas Discriminatorias.

La pregunta más importante antes de clasificar es: ¿qué atributo es mejor para clasificar? Debemos seleccionar el atributo que sea el más usado para clasificar los ejemplos. Para seleccionar este atributo vamos a definir una serie de medidas:

- Entropía.
- Ganancia de Información de Quinlan.

Para definir exactamente la ganancia de información, tenemos que empezar definiendo una medida comúnmente usada en la teoría de la información llamada Entropía (Shannon 1948) [SHA48], la cual caracteriza la pureza (impureza) de un conjunto arbitrario de ejemplos.

Dado un conjunto S, el cual contiene '+' y '-'ejemplos de una clave, la Entropía relativa a esta clasificación booleana se define como:

$$Entropía\ (S) \equiv -p_+ \log_2 p_+ - p_- \log_2 p_-$$

Dada la medida de Entropía como una medida de pureza de los ejemplos de aprendizaje, podemos definir la ganancia de información (según Quinlan), Ganancia(S, A), de un atributo A en un conjunto de ejemplos S como:

$$Ganancia\ (S, A) \equiv Entropía\ (S) - \sum_{v \in Valores\ (A)} \frac{|S_v|}{|S|} Entropía\ (S_v)$$

Donde Valores(A) es la asignación de todos los valores posibles del atributo A, y es la asignación de S para cada atributo A que tenga el valor v. S_v

O sea, la primera parte de la ecuación es la Entropía del conjunto original y la segunda es el valor esperado de la entropía después que S sea dividido usando el atributo A. Esta es el sumatorio de las Entropías de cada asignación de S_v.

Cuando utilizamos como medida la Ganancia de Información se elige clasificar (en cada nodo no hoja) según el atributo con mayor valor.

3.3.3 Caso de estudio utilizando el ID3

En esta sección se presentarán un árbol y un conjunto de reglas de decisión obtenidos utilizando el ID3, para ejemplificar su aplicación. Supongamos que queremos analizar cuáles días son convenientes para jugar al tenis basándonos en la humedad, el viento y el estado del tiempo. Los datos que se utilizarán se presentan en la siguiente tabla:

Días	Observación	Temperatura	Humedad	Viento	Jugar al Tenis
D1	Soleado	Calor	Alta	Débil	No
D2	Soleado	Calor	Alta	Fuerte	No
D3	Nublado	Calor	Alta	Débil	Si
D4	Lluvioso	Suave	Alta	Débil	Si
D5	Lluvioso	Frío	Normal	Débil	Si
D6	Lluvioso	Frío	Normal	Fuerte	No
D7	Nublado	Frío	Normal	Fuerte	Si
D8	Soleado	Suave	Alta	Débil	No
D9	Soleado	Frío	Normal	Débil	Si
D10	Lluvioso	Suave	Normal	Débil	Si
D11	Soleado	Suave	Normal	Fuerte	Si
D12	Nublado	Suave	Alta	Fuerte	Si
D13	Nublado	Calor	Normal	Débil	Si
D14	lluvioso	Suave	alta	fuerte	No

Tabla 4 – caso de estudio ID3

Fuente [MIT97]

Vamos a utilizar la Ganancia de Información como medida discriminatoria. En este momento el conjunto S= {D1, D2, D3, D4, D5, D6, D7, D8, D9, D10, D11, D12, D13, D14}.

Si se la aplicamos al atributo Viento, nos queda que:

$$Valores\ (Viento) = débil\ ,\ fuerte$$
$$S = [9+,5-]$$
$$S_{débil} = [6+,2-]$$
$$S_{fuerte} = [3+,3-]$$
$$Entropía\ (S) = 0.940$$
$$Entropía\ (S_{débil}) = 0.8112$$
$$Entropía\ (S_{fuerte}) = 1$$
$$Ganancia\ (S, Viento) \equiv Entropía\ (S) - \sum_{v \in Valores\ (A)} \frac{|S_v|}{|S|} Entropía\ (S_v) =$$
$$= Entropía\ (S) - (8/14) Entropía\ (S_{débil}) - (6/14) Entropía\ (S_{fuerte}) = 0.048$$

Si se lo aplicáramos al atributo Humedad quedaría:

$$Valores\ (Humedad\ ) = alta\ , normal$$
$$S\ = [9+,5-]$$
$$S_{alta}\ = [3+,4-]$$
$$S_{normal}\ = [6+,1-]$$
$$Entropía\ (S) = 0.940$$
$$Entropía\ (S_{alta}) = 0.985$$
$$Entropía\ (S_{normal}) = 0.591673$$
$$Ganancia\ (S, Humedad\ ) \equiv Entropía\ (S) - \sum_{v \in Valores\ (A)} \frac{|S_v|}{|S|} Entropía\ (S_v) =$$
$$= Entropía\ (S) - (7/14)\, Entropía\ (S_{alta}) - (7/14)\, Entropía\ (S_{normal}) = 0.048$$

Si se la aplicamos al atributo Temperatura:

$$Valores\ (Temperatur\ a) = calor\ , suave\ , frío$$
$$S\ = [9+,5-]$$
$$S_{calor}\ = [2+,2-]$$
$$S_{suave}\ = [4+,2-]$$
$$S_{frío}\ = [3+,1-]$$
$$Entropía\ (S) = 0.940$$
$$Entropía\ (S_{calor}) = 1$$
$$Entropía\ (S_{suave}) = 0.9182$$
$$Entropía\ (S_{frío}) = 0.811278$$
$$Ganancia\ (S, Temperatur\ a) \equiv Entropía\ (S) - \sum_{v \in Valores\ (A)} \frac{|S_v|}{|S|} Entropía\ (S_v) =$$
$$= Entropía\ (S) - (4/14)\, Entropía\ (S_{calor}) - (6/14)\, Entropía\ (S_{suave}) -$$
$$- (4/14)\, Entropía\ (S_{frío}) = 0.029$$

Si se la aplicamos al atributo Observación:

$$Valores\ (Observació\ \ n) = soleado\ \ , nublado\ \ , lluvioso$$
$$S = [9+,5-]$$
$$S_{soleado} = [2+,3-]$$
$$S_{nublado} = [4+,0-]$$
$$S_{lluvioso} = [3+,2-]$$
$$Entropía\ \ (S) = 0.940$$
$$Entropía\ \ (S_{soleado}) = 0.940$$
$$Entropía\ \ (S_{nublado}) = 0$$
$$Entropía\ \ (S_{lluvioso}) = 0.9709$$
$$Ganancia\ \ (S, Observació\ \ n) \equiv Entropía\ \ (S) - \sum_{v \in Valores\ (A)} \frac{|S_v|}{|S|} Entropía\ \ (S_v) =$$
$$= Entropía\ \ (S) - (5/14)\, Entropía\ \ (S_{soleado}) - (5/14)\, Entropía\ \ (S_{lluvioso}) = 0.2465$$

De acuerdo con los resultados, el atributo Observación es el que vamos a tomar para la raíz del árbol, y las ramas se crearán con sus posibles valores (soleado, nublado, y lluvioso).

Una vez creadas sus ramas hay que volver a escoger un atributo para cada una de ellas, repitiendo el proceso anterior. Para la que tiene el valor soleado, el conjunto quedaría así : S= {D1, D2, D8, D9, D11 }.

Para el atributo viento (rama Soleado)

$$Valores\,(Viento) = débil\,,\ fuerte$$
$$S = [2+,3-]$$
$$S_{débil} = [1+,2-]$$
$$S_{fuerte} = [1+,1-]$$
$$Entropía\,(S) = 0.970$$
$$Entropía\,(S_{débil}) = 0.9182$$
$$Entropía\,(S_{fuerte}) = 1$$
$$Ganancia\,(S, Viento) \equiv Entropía\,(S) - \sum_{v \in Valores\,(A)} \frac{|S_v|}{|S|} Entropía\,(S_v) =$$
$$= Entropía\,(S) - (3/5)\, Entropía\,(S_{débil}) - (2/5)\, Entropía\,(S_{fuerte}) = 0.01998$$

Para el atributo temperatura (rama Soleado)

$$Valores\ (Temperatur\ \ a) = calor\ ,\ suave\ ,\ frío$$
$$S = [2+,3-]$$
$$S_{calor} = [0+,2-]$$
$$S_{suave} = [1+,1-]$$
$$S_{frío} = [1+,0-]$$
$$Entropía\ (S) = 0.9709$$
$$Entropía\ (S_{calor}) = 0$$
$$Entropía\ (S_{suave}) = 1$$
$$Entropía\ (S_{frío}) = 0$$
$$Ganancia\ (S, Temperatur\ \ a) \equiv Entropía\ (S) - \sum_{v \in Valores\ (A)} \frac{|S_v|}{|S|} Entropía\ (S_v) =$$
$$= Entropía\ (S) - (2/5) Entropía\ (S_{calor}) - (2/5) Entropía\ (S_{suave}) -$$
$$- (1/5) Entropía\ (S_{frío}) = 0.5709$$

Para el atributo humedad (rama Soleado)

$$Valores\,(Humedad) = alta, normal$$
$$S = [2+,3-]$$
$$S_{alta} = [0+,3-]$$
$$S_{normal} = [2+,0-]$$
$$Entropía\,(S) = 0.970951$$
$$Entropía\,(S_{alta}) = 0$$
$$Entropía\,(S_{normal}) = 0$$
$$Ganancia\,(S, Humedad) \equiv Entropía\,(S) - \sum_{v \in Valores\,(A)} \frac{|S_v|}{|S|} Entropía\,(S_v) =$$
$$= Entropía\,(S) - (3/5) Entropía\,(S_{alta}) - (2/5) Entropía\,(S_{normal}) = 0.9709$$

Una vez que hemos terminado con la rama de soleado, tenemos que hacer los mismo con la rama lluvioso. La rama nublado solo tiene valores ´+´ así que tomaría el valor 'sí'.Ahora el conjunto sería: S = {D4, D5, D6, D10, D14 }.

Para el atributo de temperatura (rama lluvioso)

$$Valores\ (Temperatur\ a) = calor\ , suave\ , frío$$
$$S = [3+, 2-]$$
$$S_{calor} = [0+, 0-]$$
$$S_{suave} = [2+, 1-]$$
$$S_{frío} = [1+, 1-]$$
$$Entropía\ (S) = 0.9709$$
$$Entropía\ (S_{calor}) = 0$$
$$Entropía\ (S_{suave}) = 0.9182$$
$$Entropía\ (S_{frío}) = 1$$
$$Ganancia\ (S, Temperatur\ a) \equiv Entropía\ (S) - \sum_{v \in Valores\ (A)} \frac{|S_v|}{|S|} Entropía\ (S_v) =$$
$$= Entropía\ (S) - (3/5) Entropía\ (S_{suave}) - (2/5) Entropía\ (S_{frío}) = 0.0199$$

Para el atributo Viento (rama lluvioso)

$$Valores\ (Viento\ ) = débil\ , fuerte$$
$$S = [3+, 2-]$$
$$S_{débil} = [0+, 2-]$$
$$S_{fuerte} = [4+, 0-]$$
$$Entropía\ (S) = 0.970$$
$$Entropía\ (S_{débil}) = 0$$
$$Entropía\ (S_{fuerte}) = 0$$
$$Ganancia\ (S, Viento\ ) \equiv Entropía\ (S) - \sum_{v \in Valores\ (A)} \frac{|S_v|}{|S|} Entropía\ (S_v) =$$
$$= Entropía\ (S) - 0 - 0 = 0.9707$$

Para el atributo humedad (rama lluvioso)

$$Valores\ (Humedad) = alta, normal$$
$$S = [3+, 2-]$$
$$S_{alta} = [1+, 1-]$$
$$S_{normal} = [2+, 1-]$$
$$Entropía\ (S) = 0.970$$
$$Entropía\ (S_{alta}) = 1$$
$$Entropía\ (S_{normal}) = 0.8192$$
$$Ganancia\ (S, Viento) \equiv Entropía\ (S) - \sum_{v \in Valores\ (A)} \frac{|S_v|}{|S|} Entropía\ (S_v) =$$
$$= Entropía\ (S) - (2/5) * 1 - (3/5) * 0.8192 = 0.0199$$

El resultado total del árbol se presenta en la siguiente Figura. Obsérvese que en la rama de nublado todos los ejemplos están a 'sí'. Esto significa que hemos llegado a un nodo hoja con la clasificación "Jugar al tenis" = sí. Por el contrario, las otras dos ramas son no hojas, así que hay que seguir elaborando el árbol a partir de estos nodos.

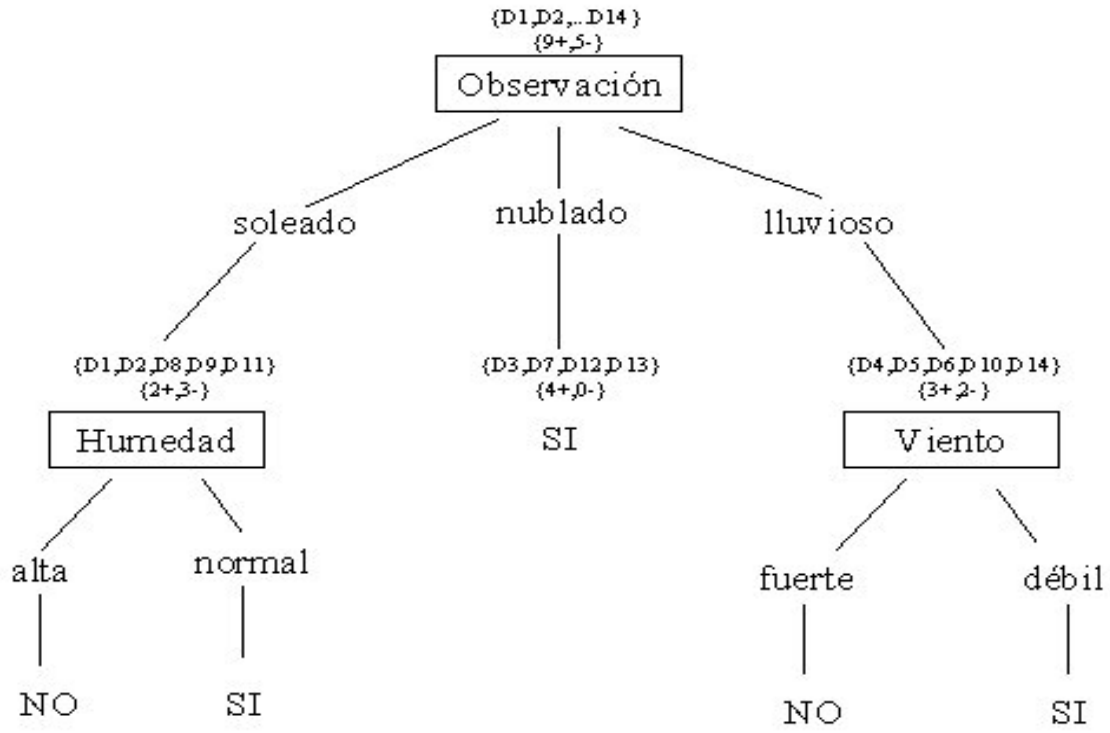


Figura 8 – Árbol ID3

Fuente [MIT97]

El proceso de selección de un nuevo atributo y particionado de los ejemplos se repite para cada nodo no hoja, usando sólo los ejemplos asociados a esa hoja. Se excluyen los que ya han salido en las ramas predecesoras. En la Figura 2, el nodo hoja denotado con `SI` significa que jugamos al tenis. El que está etiquetado con `NO` significa que no jugamos al tenis.Así se haría con cada nodo hasta que se dé una o dos de estas condiciones:

- Todo atributo esté incluido en el árbol.
- Los ejemplos asociados con el nodo hoja tienen el mismo valor en el atributo (por ejemplo, su entropía sea cero).

El conjunto de reglas que se deducen del árbol anterior es:

- Si (Observación = soleado) y (Humedad = alta) Entonces “No jugamos al tenis”.
- Si (Observación = soleado) y (Humedad = normal) Entonces “Si jugamos al tenis”.
- Si (Observación = nublado) Entonces “Si jugamos al tenis”.
- Si (Observación = lluvioso) y (viento = fuerte) Entonces “No jugamos al tenis”.
- Si (Observación = lluvioso) y (viento = débil) Entonces “Si jugamos al tenis”.

3.3.4 Conclusiones

Normalmente el algoritmo ID3 genera árboles de decisión bastante buenos en comparación con algoritmos de la familia TDIDT. Pero presenta el inconveniente de que favorece excesivamente los atributos con muchos valores posibles, y éstos no siempre son los más relevantes (la edad o el peso de una persona son atributos con valores muy diversos. sin embargo, no siempre son interesantes para hacer clasificaciones según la raza, por ejemplo).

Hasta ahora hemos supuesto que el conjunto de ejemplos del que disponíamos no contenía ni valores desconocidos ni valores erróneos. Sin embargo, esto no es cierto la mayoría de las veces, así que se pueden presentar dos problemas relacionados con el conjunto de ejemplos:

a) Atributos con valor desconocido: puede darse el caso de que cuando observamos objetos del mundo real no siempre conocemos todas sus características. Para eso podemos optar por varias opciones: tratar desconocido como un valor más, eliminar aquellos elementos que tienen algún atributo con valor desconocido, asignar al atributo el valor más frecuente entre los elementos que pertenecen a la misma clase.

b) Ruido: es cualquier información que siendo incorrecta es capaz de pasar por válida. Para tratar de eliminar el ruido existen diversos métodos que se pueden clasificar en dos grandes grupos: pre-running (mientras se construye el árbol se poda la rama no significante) y post-running (basados en estimaciones de errores).

3.4 Algoritmo Naive Bayes

Naive Bayes es uno de los clasificadores más utilizados por su simplicidad y rapidez. Se trata de una técnica de clasificación y predicción supervisada que construye modelos que predicen la probabilidad de posibles resultados. Constituye una técnica supervisada porque necesita tener ejemplos clasificados para que funcione.

Antes de empezar a describir y estudiar el algoritmo de naive bayes , explicamos algunos conceptos basicos en los cual se basa este classificador estadistico, de lo cuales el mas importante es el teorema de bayes.

3.4.1 Teorema de Bayes

El teorema de Bayes se utiliza para revisar probabilidades previamente calculadas cuando se posee nueva información. Desarrollado por el reverendo Thomas Bayes en 1763 [BAY63], el teorema de Bayes es una extensión de la probabilidad condicional.

El teorema de Bayes expresa la probabilidad condicional de un evento aleatorio A dado B en términos de la distribución de probabilidad condicional del evento B dado A y la distribución de probabilidad marginal de sólo A. En términos más generales y menos matemáticos, el teorema de Bayes es de enorme relevancia puesto que vincula la probabilidad de A dado B con la probabilidad de B dado A.

Comúnmente se inicia un análisis de probabilidades con una asignación inicial, probabilidad a priori. Cuando se tiene alguna información adicional se procede a calcular las probabilidades revisadas o a posteriori. Empezamos con dos ecuaciones de probablidad condicional, las probablidades condicionadas de A dado B , y de B dado A.

$P(A/B) = P(AnB) / P(B)$

$P(B/A) = P(BnA) / P(A)$

Despejamos de las formulas anteriores P(AnB) y P(BnA) y obtenemos lo siguiente

P(AnB) = P(A/B) x P(B)

P(BnA) = P(B/A) x P(A)

La probablidad P(AnB) = P(BnA) , por lo tanto obtenemos la siguiente ecuacion

P(A/B) x P(B) = P(B/A) x P(A)

P(A/B) = P(B/A) x P(A) / P(B)

El teorema de Bayes permite calcular las probabilidades a posteriori y es:

Sea $\{A_1, A_2, ..., A_i, ..., A_n\}$ un conjunto de sucesos mutuamente excluyentes y exhaustivos, y tales que la probabilidad de cada uno de ellos es distinta de cero (0).

Sea B un suceso cualquiera del que se conocen las probabilidades condicionales $P(B|A_i)$. Entonces, la probabilidad $P(A_i|B)$ viene dada por la expresión:

$$P(A_i/B) = \frac{P(A_i) \cdot P(B/A_i)}{P(B)}$$

Donde:
$P(A_i)$ = Probabilidad a priori
$P(B/A_i)$ = Probabilidad condicional
$P(B)$ = Probabilidad Total
$P(A_i/B)$ = Probabilidad a posteriori

Sea $\{A_1, A_2, ..., A_i, ..., A_n\}$ un sistema completo de eventos tales que la probabilidad de cada uno de ellos es distinto de cero y sea B un evento cualquiera del que se conocen las probabilidades condicionadas $P(B|A_i)$

Entonces, la probabilidad del evento B, llamada probabilidad total, se calcula empleando la siguiente fórmula:

$$P(B) = P(A_1) \cdot P(B/A_1) + P(A_2) \cdot P(B/A_2) + \cdots P(A_n) \cdot P(B/A_n)$$

Con base en la definición anterior obtenemos la Fórmula de Bayes, también conocida como la Regla de Bayes

$$P(A_i|B) - \frac{P(B|A_i)P(A_i)}{\sum_{k=1}^{n} P(B|A_k)P(A_k)}$$

3.4.2 Clasificador Naive Bayes

Es el modelo de red bayesiana orientada a clasificación más simple, supone que todos los atributos son independientes conocida la variable clase.

$$v_{MAP} = argmax_{v_j \in V} \left(\frac{P(a_1,\ldots,a_n|v_j)P(v_j)}{P(a_1,\ldots,a_n)} \right)$$

$$= argmax_{v_j \in V} \left(P(a_1,\ldots,a_n \mid v_j)P(v_j) \right)$$

3.4.3 Caso de estudio utilizando Clasificador Naive Bayes

Utilizamos el mismo ejemplo usado para el algoritmo ID3, a partir de una serie de instancias o datos que tenemos, supongamos que queremos analizar cuáles días son convenientes para jugar al tenis basándonos en la humedad, el viento y el estado del tiempo. Los datos que se utilizarán se presentan en la siguiente tabla:

Días	Observación	Temperatura	Humedad	Viento	Jugar al Tenis
D1	Soleado	Calor	Alta	Débil	No
D2	Soleado	Calor	Alta	Fuerte	No
D3	Nublado	Calor	Alta	Débil	Si
D4	Lluvioso	Suave	Alta	Débil	Si
D5	Lluvioso	Frío	Normal	Débil	Si
D6	Lluvioso	Frío	Normal	Fuerte	No
D7	Nublado	Frío	Normal	Fuerte	Si
D8	Soleado	Suave	Alta	Débil	No
D9	Soleado	Frío	Normal	Débil	Si
D10	Lluvioso	Suave	Normal	Débil	Si
D11	Soleado	Suave	Normal	Fuerte	Si
D12	Nublado	Suave	Alta	Fuerte	Si
D13	Nublado	Calor	Normal	Débil	Si
D14	lluvioso	Suave	alta	fuerte	No

Tabla 5 – Caso de estudio Naive Bayes

Fuente [MIT97]

La instancia a clasificar será la siguiente,

Observación: *soleado****, Temperatura:*** *Fría****, Humedad:*** *Alta****, Viento:*** *Fuerte*

A partir de esta instancia iremos calculando si un día con estas condiciones de tiempo es conveniente para jugar al tenis

Calculamos las probabilidades individuales P (Jugar= Si) = 9/14 = 0.64

De la misma forma calculamos la probabilidad P (jugar=No) = 5/14 = 0.36

Días	Observación	Temperatura	Humedad	Viento	Jugar al Tenis
D1	Soleado	Calor	Alta	Débil	No
D2	Soleado	Calor	Alta	Fuerte	No
D3	Nublado	Calor	Alta	Débil	Si
D4	Lluvioso	Suave	Alta	Débil	Si
D5	Lluvioso	Frío	Normal	Débil	Si
D6	Lluvioso	Frío	Normal	Fuerte	No
D7	Nublado	Frío	Normal	Fuerte	Si
D8	Soleado	Suave	Alta	Débil	No
D9	Soleado	Frío	Normal	Débil	Si
D10	Lluvioso	Suave	Normal	Débil	Si
D11	Soleado	Suave	Normal	Fuerte	Si
D12	Nublado	Suave	Alta	Fuerte	Si
D13	Nublado	Calor	Normal	Débil	Si
D14	lluvioso	Suave	alta	fuerte	No

Tabla 6 – Caso de estudio Naive Bayes con resultado positivo

De igual forma calculamos las probabilidades condicionadas

P (Observación= soleado / Jugar= SI) = 2/9 = 0.22

Días	Observación	Temperatura	Humedad	Viento	Jugar al Tenis
D1	Soleado	Calor	Alta	Débil	No
D2	Soleado	Calor	Alta	Fuerte	No
D3	Nublado	Calor	Alta	Débil	Si
D4	Lluvioso	Suave	Alta	Débil	Si
D5	Lluvioso	Frío	Normal	Débil	Si
D6	Lluvioso	Frío	Normal	Fuerte	No
D7	Nublado	Frío	Normal	Fuerte	Si
D8	Soleado	Suave	Alta	Débil	No
D9	Soleado	Frío	Normal	Débil	Si
D10	Lluvioso	Suave	Normal	Débil	Si
D11	Soleado	Suave	Normal	Fuerte	Si
D12	Nublado	Suave	Alta	Fuerte	Si
D13	Nublado	Calor	Normal	Débil	Si
D14	lluvioso	Suave	alta	fuerte	No

Tabla 7 – Caso de estudio Naive Bayes, probabilidad jugar con tiempo soleado

De la misma forma calculamos las probabilidades restantes

P (Observación =soleado / jugar=no) = 3/5 = 0,6

P (Temperatura =fría / jugar=si) = 3/9 = 0,33

P (Temperatura=fría / jugar=no) = 1/5 = 0,2

P (Humedad=alta / jugar=si) = 3/9 = 0,33

P (humedad=alta / jugar=no) = 4/5 = 0,8

P (Viento=cierto / jugar=si) = 3/9 = 0,33

P (Viento=cierto / jugar=no) = 3/5 = 0,6

La clave para la clasificación Naive Bayes es calcular recuentos en conjunto. En el ejemplo de demostración, hay cuatro valores de atributo independientes (Observación, Temperatura, Humedad, Viento) y dos valores de atributo dependientes (jugar=SI, Jugar=NO), así que se deben calcular y almacenar un total de 4 * 2 = 8 recuentos en conjunto. Aplicando la formula anterior del clasificador a nuestro caso particular obtenemos el siguiente resultado

$$v_{NB} = \arg\max_{v_j \in V} p(v_j) \prod_i p(a_i / v_j)$$

$$= \arg\max_{v_j \in V} p(v_j)\, p(Cielo = soleado / v_j)\, p(Temp = fría / v_j)$$

$$p(Humedad = alta / v_j) p(Viento = cierto / v_j)$$

$$p(si)\, p(soleado / si)\, p(fría / si)\, p(alta / si) p(cierto / si) = 0{,}0053 \quad (0{,}205)$$

$$p(no)\, p(soleado / no)\, p(fría / no)\, p(alta / no) p(cierto / no) = 0{,}0206 \quad (0{,}795)$$

Por la tanto y según el cálculo anterior la respuesta seria NO, un día con estas condiciones atmosféricas no es conveniente para jugar al tenis

3.4.4 Conclusiones

Naive Bayes Es uno de los algoritmos de aprendizaje más prácticos, junto a árboles, redes de neuronas y K-NN, necesita un conjunto de entrenamiento grande es decir un conjunto grande de instancias y que los atributos sean razonablemente independientes, entre las aplicaciones más importantes de este clasificador están la Diagnosis y la Clasificación de texto.

La clasificación de Naive Bayes es una técnica con una tecnología de aprendizaje que se puede usar para predecir a qué categoría en especial pertenece determinado caso de datos , es un algoritmo supervisado tal que necesitamos conocer la clase de los ejemplos para estimar la probabilidad a posteriori de las observaciones, tiene un enfoque probabilístico al aprendizaje, se puede considerar

que es un algoritmo competitivo las hipótesis compiten entre sí, *venciendo* la que tenga mayor probabilidad.

Se puede afirma sin miedo a equivocarnos que algunos métodos bayesianos se encuentran entre los más eficiente y que nos permiten interpretar el funcionamiento de otros métodos en términos probabilísticos, Incluso cuando no son aplicables, proporcionan un estándar de toma de decisión óptima, frente al que comparar otros métodos.

La clasificación de Naive Bayes es una técnica relativamente rudimentaria, pero tiene varias ventajas sobre otras alternativas más sofisticadas, como la clasificación de redes neuronales, la clasificación de regresión logística y la clasificación de máquina de vectores de soporte. Naive Bayes es sencillo, relativamente fácil de implementar y escala bien a grandes conjuntos de datos. Además, Naive Bayes se extiende fácilmente a problemas de clasificación multinominales, aquellos con tres o más variables dependientes.

Alguna de las Ventajas del algoritmo de Naive Bayes es la facilidad de implementación además obtiene buenos resultados en gran parte de los casos.

Por otra parte también tiene una serie de desventajas de las cuales Asumir que las variables tienen independencia condicional respecto a la clase lleva a una falta de precisión, Ya que en la práctica, existen dependencias entre las variable por ejemplo.: en datos hospitalarios: Perfil (edad, historia familiar, etc.), Síntomas (fiebre, tos, etc.) Enfermedad (cáncer de pulmón, diabetes, etc.), Con un clasificador Naïve Bayes no se pueden modelar estas dependencias.

3.5 Algoritmo K-Means

El algoritmo K-means, creado por MacQueen en 1967 según [MAC67]. Es el algoritmo de clustering particional más conocido y utilizado ya que es de muy simple aplicación y eficaz. Sigue un procedimiento simple de clasificación de un conjunto de objetos en un determinado número K de clústeres, el número K será determinado a priori, Esta destinado a situaciones en las cuales todas las variables son del tipo cuantitativo, y la distancia cuadrática Euclidea es elegida como medida de diferencia.

se puede considerar un algoritmo basado en distancias, ya que se calcula la distancia de cada objeto al centro del centroide o al clúster, La distancia que se utiliza en el algoritmo para medir las distancias entre los elementos es la distancia euclídea.

El nombre de k-means (k-medias) viene porque representa cada uno de los clústeres por la media (o media ponderada) de sus puntos, es decir, por su centroide. La representación mediante centroides tiene la ventaja de que tiene un significado gráfico y estadístico inmediato.

El término "k-means" fue utilizado por primera vez por James MacQueen en 1967, aunque la idea se remonta a Hugo Steinhaus en 1957 según [STE57].

3.5.1 Descripción del algoritmo

Dado un conjunto de observaciones (x1, x2,…, xn), donde cada observación es un vector real de D dimensiones, k-means construye a una partición de las observaciones en k conjuntos (k = n) → S = {S1, S2,…, Sk}

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x}_j \subset S_i} \|\mathbf{x}_j - \boldsymbol{\mu}_i\|^2$$

Donde μi es la media de puntos en Si.

3.5.2 Etapas del algoritmo

El algoritmo del K-means se realiza en 4 etapas:

Paso 1: la primera etapa es elegir aleatoriamente un número K, ese número representa el número de clústeres iniciales, Se consideran los n primeros elementos de la matriz de datos de entrada como k conglomerados o centroides con un único elemento, con la particularidad de que se van a permutar estos elementos para que el inicio de los primeros centroides sea diferente cada vez.

Paso 2: Asignar en el orden de la matriz de datos cada uno de los objetos al centroide mas próximo, después de cada asignación se recalculara el nuevo centroide.

Paso 3: Después de que todos los objetos hayan sido asignados en el paso anterior, calcular los centroides de los conglomerados obtenidos, y reasignar cada objeto al centroide más cercano.

Pasó 4: Repetir los pasos 2 y 3 hasta que se alcance un determinado criterio de parada.

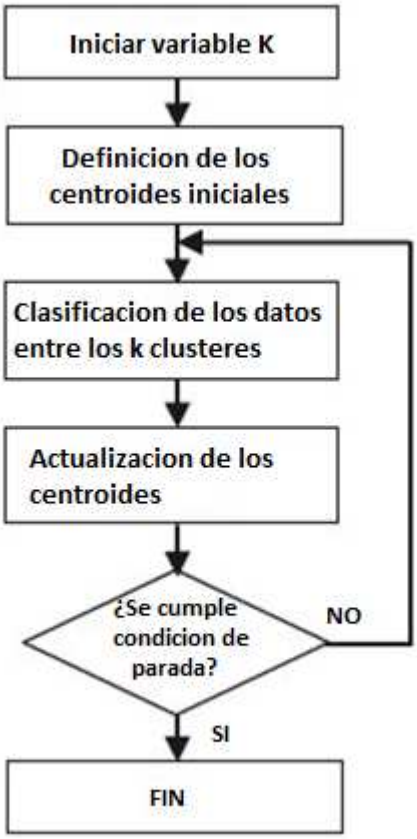


Figura 9 – Diagrama Flujo de KMEANS

En el algoritmo propuesto por MacQueen comienza considerando los n primeros elementos del Fichero de casos o matriz de datos como los k centroides iniciales, o dicho de forma equivalente como conglomerados o centroides con un único elemento.

Otro posible método para inicializar el algoritmo es tomar k observaciones de muestra completamente al azar y dichas observaciones se convierten en los k centroides iniciales.

A continuación, y siguiendo el orden establecido en el fichero, cada uno de los objetos se va asignando al conglomerado con centroide más próximo, con la característica de que al efectuar cada asignación se recalculan las coordenadas del nuevo centroide.

Finalmente, una vez asignados todos los objetos, se calculan los centroides para cada uno de los conglomerados y se reasigna cada objeto al centroide más cercano sin que en este paso se lleve a cabo una re calculación del centroide para cada asignación. Los pasos anteriores se iteran hasta que se verifique un determinado criterio de parada.

El criterio de parada puede ser uno de los siguientes:

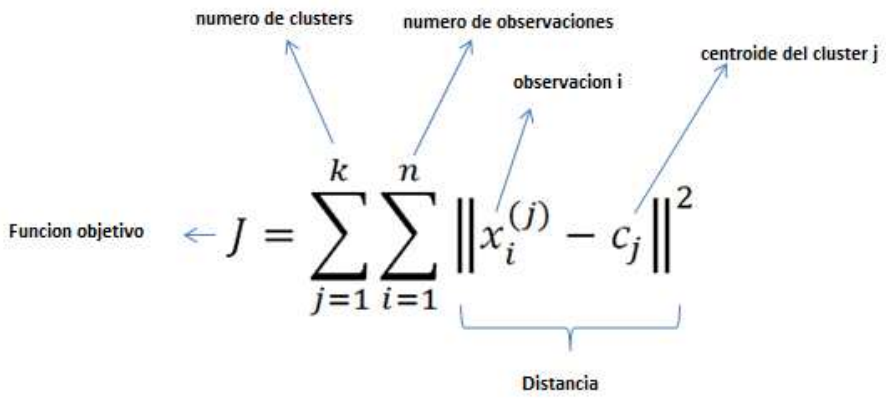
- Un número máximo de iteraciones
- cuando las asignaciones ya no cambian.
- El que los nuevos centroides disten de los centroides obtenidos en la iteración previa menos que una determinada distancia, etc...

Habitualmente el algoritmo K-means converge a las pocas iteraciones, El algoritmo se considera que ha convergido cuando las asignaciones ya no cambian.

3.5.3 Objetivo del algoritmo K-Means

La suma de las discrepancias entre un punto y su centroide, expresado a través de la distancia apropiada, se usa como función objetivo. La función objetivo, suma de los cuadrados de los errores entre los puntos y sus centroides respectivos, es igual a la varianza total dentro del propio clúster (la varianza intra-cluster).

El objetivo de la agrupación K-Means es minimizar la varianza intra-cluster total, es decir, la siguiente función objetivo:



El algoritmo intenta encontrar la mejor asignación de puntos a los distintos grupos o clústeres que habíamos definido anteriormente, la mejor asignación en el sentido de maximizar las distancias entre los distintos clústeres(Inter-clúster) y minimizar las distancias dentro de un clúster(Intra-clúster).

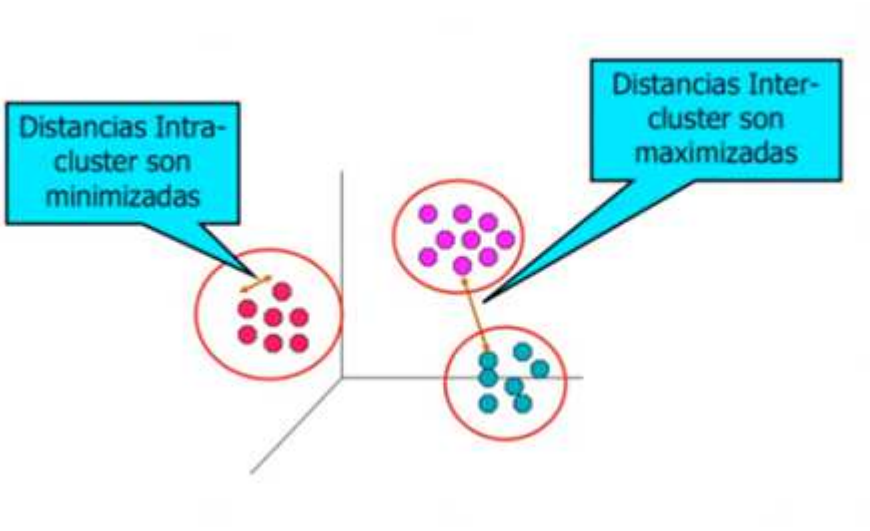


Figura 10 – Distancias Intra-clúster y inter-clúster

Fuente: [PAN06]

Es importante tener en cuenta que el algoritmo de McQueen es sensible al orden, con el que se encuentran los objetos en el Fichero de casos de entrada o la matriz de datos, y fundamentalmente es sensible a los objetos que se encuentran en las K primeras posiciones, lo cual supone una desventaja para el algoritmo.

3.5.4 Caso de Estudio

Esta sección se explica la resolución de un sencillo ejemplo usando K-Means, para eso tomamos un conjunto de datos o observaciones, en este caso cada tupla de datos o observación tendrá dos variables altura, peso y ejecutaremos el algoritmo de forma iterativa hasta obtener un resultado óptimo que verifique un criterio de parada dado de antemano.

Supongamos que queremos obtener 3 grupos a partir de 9 objetos representados en la tabla y figuras anteriores, por lo tanto el valor de K es igual a 3, tendremos 3 clústeres de datos.

Observación	Altura (cm)	Peso (kg)
A	180	90
B	175	75
C	185	70
D	181	95
E	174	65
F	175	90
G	169	65
H	192	78
J	186	75

Tabla 8 – Caso de estudio algoritmo KMEANS

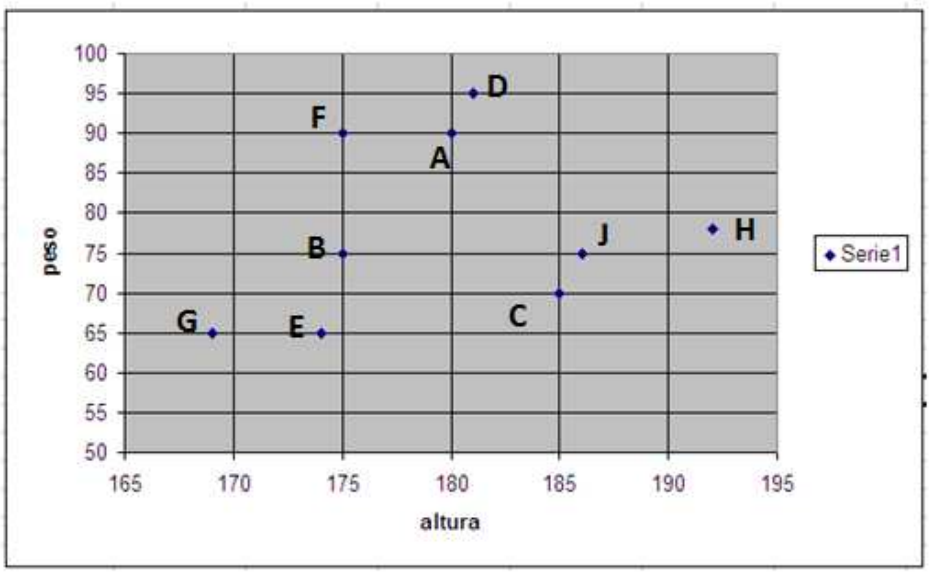


Figura 11 – Representación caso de estudio algoritmo KMEANS

Seleccionamos los 3 primeros elementos de la tabla de datos, estos 3 primeros elementos A, B, C serán los 3 centroides iniciales, aquí obtenemos 3 conglomerados o centroides con un único elemento:

- Cluster1 { A } , centroide A
- Cluster2 { B } , centroide B
- Cluster3 { C } , centroide C

Ahora iremos asignando el resto de elementos según el orden en la tabla,
Obtenemos que el elemento D se integra en el clúster con centroide A, después de la asignación recalculamos las coordenadas del nuevo centroide, el punto A deja de ser el centroide del cluster1 y pasa a ser el punto C1 con sus nuevas coordenadas.

- Cluster1 { A , D} con C1 como centroide del cluster1

Obtenemos que el elemento E se integra en el clúster con centroide B, después de la asignación recalculamos las coordenadas del nuevo centroide, el punto B deja de ser el centroide del cluster2 y pasa a ser el punto C2 con sus nuevas coordenadas.

- Cluster2 { B, E} con C2 como centroide del cluster2

Obtenemos que el elemento F se integra en el clúster con centroide C1, después de la asignación recalculamos las coordenadas del nuevo centroide, el punto C1 ya tiene nuevas coordenadas.

- Cluster1 { A, D, F} con C1 como centroide del cluster1

Y así sucesivamente para todos los elementos de la tabla primero la asignación de la observación al centroide más cercano y después se recalculan las coordenadas del nuevo centroide.

Finalmente una vez hayamos asignado todos los elementos a un clúster, calcular los centroides de los conglomerados obtenidos, y reasignar cada objeto al centroide más cercano, al finalizar la iteración 1 obtenemos:

- Cluster1 { A, D, F } con C1 como centroide del clúster
- Cluster2 { B, E, G } con C2 como centroide del clúster
- Cluster3 { C, J, H } con C3 como centroide del clúster

En este ejemplo solo hizo falta una iteración para llegar a alcanzar el criterio de parada, con una iteración las asignaciones ya no cambian y por lo tanto hemos alcanzado el criterio de parada.

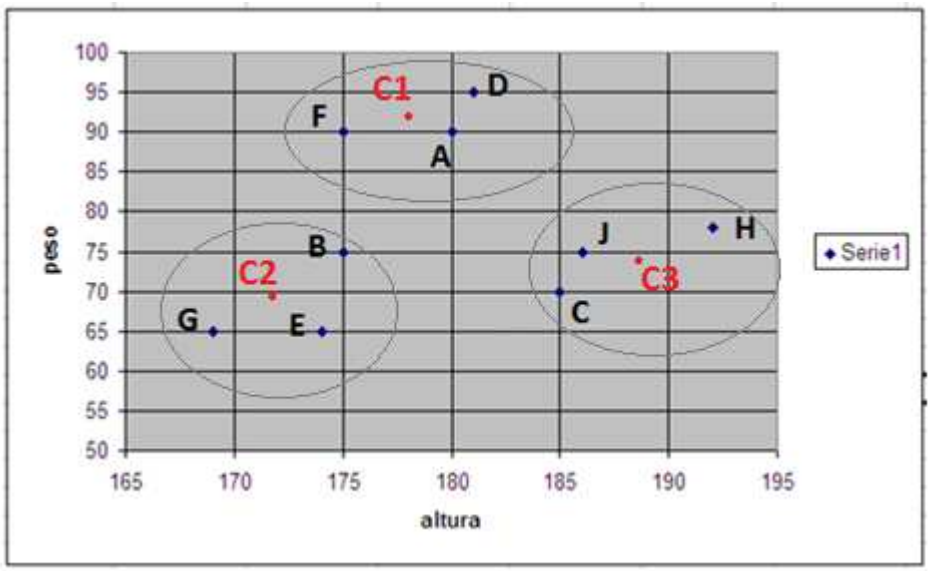


Figura 12 – Resultado caso de estudio algoritmo KMEANS

3.5.5 Conclusiones

K-Means es un algoritmo sencillo y eficiente, como existe un conjunto de entrenamiento con la clasificación de objetos preestablecida en el mismo, la clasificación de nuevos objetos sólo implica la medida de la distancia entre objetos y no requiere de un cálculo iterativo, debido a que se cuenta con más información inicial, la clasificación debería ser más exacta, si las clases iniciales impuestas son las correctas.

Uno de los inconvenientes principales del K-Means, además del hecho de que sea necesario realizar en sucesivas ocasiones el algoritmo para así tener el resultado más óptimo posible, es la necesidad de inicializar el número de prototipos al principio de la ejecución.

Es decir inicializar el número de grupos K, una elección inapropiada puede acarrear malos resultados, Por eso es muy importante cuando ejecutemos el algoritmo k-Means tener en cuenta la importancia de determinar el números de grupos para un conjunto de datos.

Esto perjudica la eficacia del algoritmo ya que en la práctica, no se conoce a priori el número de clústeres inicial. Este defecto le perjudicara al compararlo con otros algoritmos, ya que en muchos la

inicialización del número de clústeres no es necesaria. Además k-Means es susceptible a valores extremos porque distorsionan la distribución de los datos y La convergencia a óptimos locales puede traer malos resultados.

3.6 Algoritmo EM (Esperanza-Maximizacion)

El algoritmo EM (Expectation- Maximization) se engloba dentro de los métodos de aprendizaje no supervisados. Esto quiere decir que los datos que poseemos son insuficientes para realizar una estimación directa de verosimilitud, ya que desconocemos a que clase pertenece cada dato, EM pertenece a una familia de modelos que se conocen como Finite Mixture Models, los cuales se pueden utilizar para segmentar conjuntos de datos. Es un método de clustering probabilístico.

El algoritmo EM fue expuesto por Arthur Dempster, Nan Laird y Donald Rubin de la Royal Statistical Society en una publicación de 1977 [DEM77], se usa en estadística para encontrar estimadores de máxima verosimilitud de parámetros en modelos probabilísticos que dependen de variables no observables.

A continuacion describiremos el algoritmo EM, para lograr esto comenzaremos dando una breve introduccion a algunos conceptos basicos en los cuales se basa este algortimo de lo cuales los mas importantes son el concepto de estimacion de maxima verosimilitud que usaremos como criterio de calidad del agrupamiento del algoritmo y el modelo de mezclas finitas (Finite Mixture Models) los cuales nos ayudaran a comrender mejor el algoritmo EM.

3.6.1 Estimación de Máxima Verosimilitud (MLE)

Es un método de estimación supervisado, es decir, requiere de muestras etiquetadas para poder ser aplicado. MLE (Maximum Likelihood Estimation) es el método más usado para estimar distribuciones por su gran eficiencia y suele utilizarse con distrubuciones paraemtrizadas.

Su meta es encontrar el conjunto de parámetros que maximizan la probabilidad una clase de haber generado los datos que le son propios, es decir, la verosimilitud de las observaciones que pertenezcan a una clase p (X/Wk).

Sea x={X1,X2... XT } un conjunto de observaciones estadísticamente independientes que sabemos pertenecen a la clase k. Si llamamos Θk al conjunto de parámetros que determinan la distribución asociada a la clase k, la MLE viene dada por:

$$\theta^{k}_{MLE} = \underset{\theta_k}{argmax} \prod_{\forall t} p(x_t/\theta_k) = \underset{\theta_k}{argmax} \sum_{\forall t} \ln p(x_t/\theta_k)$$

Para aplicarlo a una distribución concreta sólo hay que sustituir p (x / Θk) por su valor y derivar respecto a algún parámetro.

3.6.2 Modelo de mezclas finitas

Un modelo de mezclas finitas es un modelo probabilístico para representar la presencia de sub poblaciones dentro de la población general, describe un conjunto de k grupos o clústeres,

Los clústeres no tienen la misma probabilidad, por cada uno de los clústeres o grupos, tenemos una distribución gaussiana o normal (con diferentes medias y varianzas), que refleja sus poblaciones relativas, cada distribución muestra la probabilidad de que una instancia particular tenga un cierto conjunto de valores para sus atributos si se supiera que pertenece a ese clúster, por lo tanto tenemos un conjunto de k distribuciones de probabilidad, cada una de esas distribuciones refleja las poblaciones relativas.

Para describir completamente el modelo de mezclas finitas hay que determinar 5 parámetros

- μA promedio del grupo A
- σA la desviación estándar del grupo A
- P(A) probabilidad del grupo A
- μB promedio del grupo B
- σB desviación estándar del grupo B

En el siguiente grafico vemos la descripción de un modelo de mezclas finitas sencillo, es un modelo de que tiene dos grupos o clústeres A y B, con una distribución gaussiana o normal para cada grupo, que describen un solo atributo numérico,

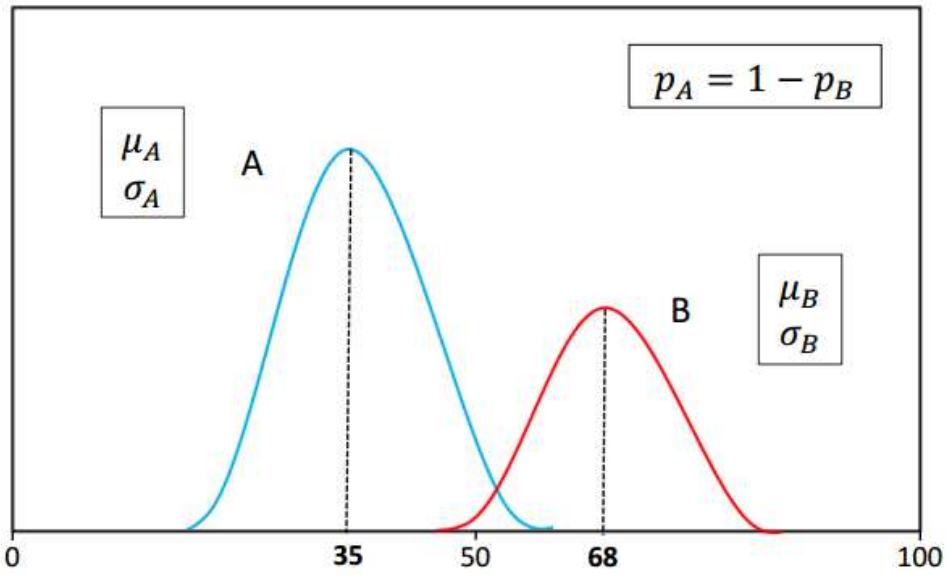


Figura 13 – Modelo de mezclas finitas

Basado en fuente: [HAL11]

WAi = Probabilidad de que el dato i pertenezca al grupo A

Xi = Dato i

Promedio del grupo A

$$\mu_A = \frac{w_{A1}x_1 + w_{A2}x_2 + \cdots + w_{An}x_n}{w_{A1} + w_{A2} + \cdots + w_{An}}$$

Varianza del grupo A

$$\sigma_A^2 = \frac{w_{A1}(x_{A1} - \mu_A)^2 + w_{A2}(x_{A2} - \mu_A)^2 + \cdots + w_{An}(x_{An} - \mu_A)^2}{w_{A1} + w_{A2} + \cdots + w_{An}}$$

Probabilidad del grupo A

$$p_A = \frac{w_{A1} + w_{A2} + \cdots + w_{An}}{(w_{A1} + w_{A2} + \cdots + w_{An}) + (w_{B1} + w_{B2} + \cdots + w_{Bn})}$$

Función normal

$$f(x;\ \mu_A, \sigma_A) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Probabilidad de que x pertenece al grupo A

$$p_{Ax} = \frac{f(x;\ \mu_A, \sigma_A)p_A}{f(x;\ \mu_A, \sigma_A)p_A + f(x;\ \mu_B, \sigma_B)p_B}$$

Likelihood (función de verosimilitud)

$$\prod_i (p_A P(x_i|A) + p_B P(x_i|B))$$

Log de likelihood (log-likelihood):

$$LP = \ln\left(\prod_i (p_A P(x_i|A) + p_B P(x_i|B))\right)$$

$$LP = \sum_i \ln\big((p_A P(x_i|A) + p_B P(x_i|B))\big)$$

3.6.3 El algoritmo EM

es un método ampliamente aplicado en la computación iterativa de la estimación de máxima verosimilitud (MLE),. En cada iteración del algoritmo EM, hay dos pasos :

- paso-E o el paso de esperanza (o expectación) : partiendo de un número de clases dado, cada una con sus valores para los parámetros, calculamos la verosimilitud de las observaciones respecto a cada clase.
- paso-M o el paso de maximización: usando el valor de esa verosimilitud para ponderar la pertenencia a cada clase de las observaciones, reestimamos los parámetros de las clases según MLE. Y vuelta a empezar, hasta que la verosimilitud deje de variar

La situación en la que el algoritmo EM muestra toda su potencia es en los problemas de datos incompletos, donde la estimación de máxima verosimilitud resulta difícil debido a la ausencia de alguna parte de los datos dentro de una estructura de datos simple y familiar.

El algoritmo EM está estrechamente relacionado con el método ad-hoc para la estimación con datos perdidos, donde los parámetros son estimados después de haber imputado los datos perdidos con valores determinados al inicio. Estos valores de imputados son actualizados por sus valores predichos usando estos estimadores iniciales de los parámetros. Los parámetros son entonces reestimados, y así sucesivamente, procediendo iterativamente hasta la convergencia.

Puede verse en cierta forma similar al algoritmo k-means, En el algoritmo k-means escogemos unos centroides iniciales, y mediante algún tipo de distancia asignamos cada observación al centroide más cercano, para luego recalcular los centroides con las observaciones que han sido clasificadas como pertenecientes a un centroide, obteniendo nuevas coordenadas del centroide.

El algoritmo EM es similar,salvo que la medida de distancia es la verosimilitud p (X/Wi) y depende por tanto de la distribución usada para modelar cada clase, y la asignación de las observaciones a las distintas clases es blanda, no del tipo todo o nada que se utiliza en k-means.

3.6.4 Criterio de parada o terminación

El algoritmo converge a un máximo local, pero nunca alcanza dicho máximo, el resultado será más fiable cuanto más cerca este de ese máximo local. Es una medida de la calidad del agrupamiento, dicha calidad aumenta con cada iteración del algoritmo, Ese criterio de calidad se conoce como el likelihood de los datos, se trataría entones de estimar los parámetros buscados maximizando ese likelihood (ese criterio se conoce como MLE definido anteriormente), su magnitud refleja la calidad del agrupamiento.

En la práctica nos vale con calcular el log del likelihood (log-likelihood), Durante una primera parte del aprendizaje, la función de verosimilitud del conjunto de entrenamiento irá aumentando. Sin embargo, en una determinada época del entrenamiento, la función de verosimilitud del conjunto de

entrenamiento seguirá, aumentando pero el cambio será cada vez menor a medida que se vayan incrementado las iteraciones, por la tanto con cada iteración la función de verosimilitud (MLE) solo puede aumentar o permanecer constante. Lo ideal sería esperar a que la función de verosimilitud permaneciese constante, ya que eso significaría que nos encontraríamos en un máximo de la misma.

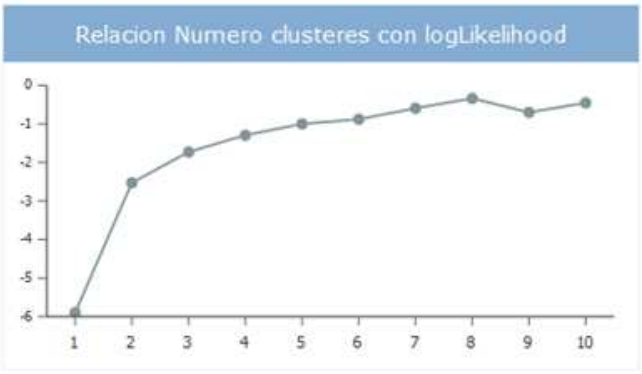


Tabla evaluacion

Numero clusteres	logLikelihood	Numero Iteraciones
1	-5,8964	2
2	-2,5283	2
3	-1,7303	4
4	-1,2934	5
5	-1,0022	16
6	-0,8797	4
7	-0,5930	2
8	-0,3390	4
9	-0,7012	2
10	-0,4544	2

Figura 14 – log likelihood EM

Sin embargo, en general nos interesará detener la simulación antes de este momento porque no nos interesa iterar demasiado si las diferencias en el resultado final van a ser mínimas. Con lo cual vamos a Iterar hasta que el aumento en el log-likelihood sea despreciable, o mejor aun Iterar hasta que la diferencia entre valores sucesivos de log-likelihood sea menor a un umbral predefinido para n iteraciones sucesivas.

3.6.5 Caso de estudio

Para nuestro caso de estudio escogeremos un grupo de personas los cuales queremos clasificar ese grupo en dos sub-grupos o clústeres, altos y bajos. Con lo que tendríamos una distribución normal por cada grupo. Tendremos un único atributo numérico que sería la altura de una persona, Este ejemplo sencillo vamos a basarnos en modelo estadístico de mezclas finitas, tenemos las instancias u observaciones pero no sabemos a qué grupo pertenece cada una de las instancias.

Partimos de la siguiente tabla donde tenemos 10 instancias o observaciones que corresponden a 10 personas.

Numero	Altura
1	195
2	166
3	188
4	195
5	179
6	198
7	161
8	179
9	200
10	191

Tabla 9 – Caso de estudio algoritmo EM

Inicializar

El primero paso es inicializar las probabilidades de pertenecía de cada instancia a uno de los grupos, se hará dicha inicialización aleatoriamente.

Numero	Altura	P(Ax)	P(Bx)	Grupo
1	195	0.86	0.14	A
2	166	0.42	0.58	B
3	188	0.52	0.48	A
4	195	0.52	0.48	A
5	179	0.95	0.05	A
6	198	0.91	0.09	A
7	161	0.51	0.49	A
8	179	0.25	0.75	B
9	200	0.87	0.13	A
10	191	0.6	0.4	A

Tabla 10 – Caso de estudio algoritmo EM inicialización aleatoriamente

Paso M

En este paso calculamos los parámetros de las funciones de distribución de probabilidad de cada clúster, Y maximizar la probabilidad de las distribuciones dados los datos disponibles.

Después de inicializar calculamos las ecuaciones que describen el modelo de mezclas finitas (μA, σA, μB, P(A), μB, σB),

Promedio del grupo A

$$\mu_A = \frac{w_{A1}x_1 + w_{A2}x_2 + \cdots + w_{An}x_n}{w_{A1} + w_{A2} + \cdots + w_{An}}$$

Varianza del grupo A

$$\sigma_A^2 = \frac{w_{A1}(x_{A1} - \mu_A)^2 + w_{A2}(x_{A2} - \mu_A)^2 + \cdots + w_{An}(x_{An} - \mu_A)^2}{w_{A1} + w_{A2} + \cdots + w_{An}}$$

Y así sucesivamente para las demás ecuaciones que están descritas en el apartado de modelo de mezclas finitas.

μA = 187,0 σA = 12,4

μB = 179,2 σB = 12,8

P (A) =0.64 P (B) = 1 - P(A) = 0.36

Paso E

En este paso se calculan las probabilidades de pertenencia a cada clúster para cada instancia, es decir la probabilidad de que un dato pertenezca a un determinado grupo.

Es decir la probabilidad de x pertenezca al grupo A:

$$p_{Ax} = \frac{f(x;\ \mu_A, \sigma_A)p_A}{f(x;\ \mu_A, \sigma_A)p_A + f(x;\ \mu_B, \sigma_B)p_B}$$

Para ello necesitamos Calcular la función de distribución normal del clúster A

$$f(x;\ \mu_A, \sigma_A) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Con esto calculamos P (Ax) y P (Bx) es decir la probabilidad de que una instancia x pertenezca al grupo A o al grupo B, siendo P (BX) = 1- P (Ax).

Calculando para todas las instancias x tenemos la siguiente tabla con las probabilidades actualizadas.

Numero	Altura	P(Ax)	P(Bx)	Grupo
1	195	0.74	0.26	A
2	166	0.44	0.56	B
3	188	0.68	0.32	A
4	195	0.74	0.26	A
5	179	0.60	0.40	A
6	198	0.76	0.24	A
7	161	0.37	0.63	B
8	179	0.60	0.40	A
9	200	0.77	0.23	A
10	191	0.71	0.29	A

Tabla 11 – Resultado caso de estudio algoritmo EM

Después comprobamos el Criterio de parada o terminación si se cumple paramos de iterar y si no, seguimos iterando el algoritmo hasta cumpliese dicho criterio.

Se trataría entones de estimar los parámetros buscados maximizando log-likelihood o la función de verosimilitud que en nuestro caso de modelo de mezclas finitas es la siguiente:

$$LP = \sum_i \ln\big((p_A P(x_i|A) + p_B P(x_i|B))\big)$$

Calculando la formula anterior obtenemos LP = -17.22, seguimos iterando con el mismo proceso anterior y vemos que con cada iteración ese likelihood va aumentando.

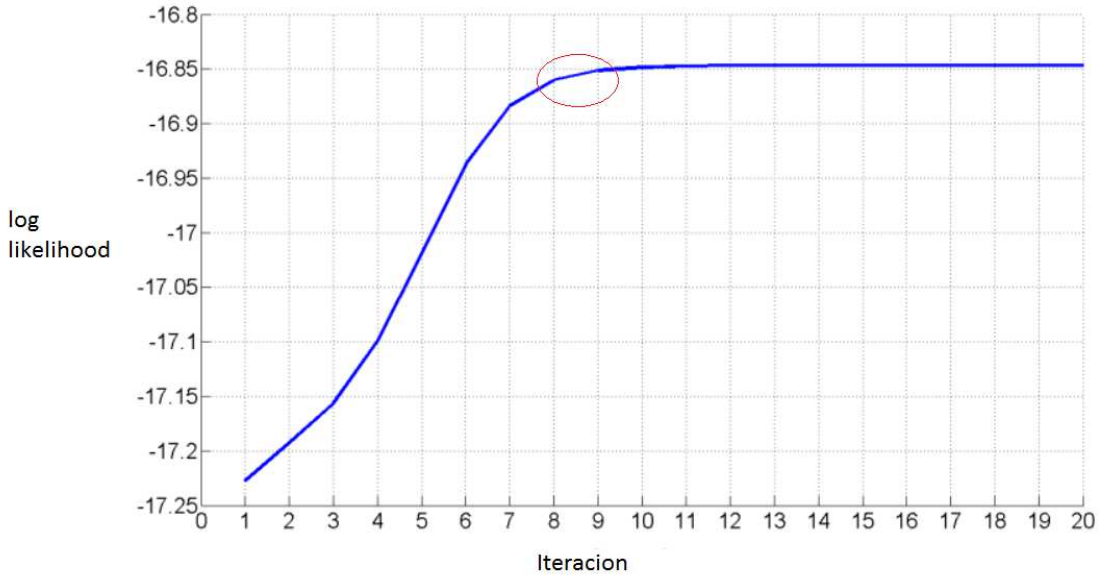


Figura 15 – log likelihood caso de estudio EM

Se puede apreciar en la grafica anterior, que a medida que incrementamos el numero de iteraciones el likelihood va aumentando hasta llegar a estabilizarse, a partir de la octava o novena iteración los cambios ya son inapreciables, por lo tanto podemos parar de iterar llegado ese punto, los parámetros hallados en ese punto o iteración serian los que maximizan ese likelihood, y por lo tanto cumplen nuestro criterio de parada o terminación.

3.6.6 Aplicaciones del algoritmo

Las situaciones donde el algoritmo EM puede ser aplicado no incluyen solamente situaciones con datos incompletos, sino también una amplia variedad de situaciones donde la presencia de datos incompletos no es del todo natural o evidente.

Éstas incluyen modelos estadísticos como efectos aleatorios, mezclas, convoluciones, modelos linearles logarítmicos, etc. Problemas de estimación hasta ahora intratables para estas situaciones han sido resueltos o se han simplificado procedimientos complicados de estimación de la verosimilitud usando el algoritmo EM.

El algoritmo EM tiene, por lo tanto, aplicaciones en casi todos los contextos estadísticos y en casi todos los campos donde se hayan aplicado técnicas estadísticas: tratamiento de imágenes médicas, corrección de censos, epidemiología del SIDA, y entrenamiento de redes neuronales artificiales, entre otros.

3.6.7 Ventajas e inconvenientes

El algoritmo EM tiene un gran potencial del algoritmo como una herramienta útil en problemas de estimación estadística, en este apartado mencionaremos tanto las ventajas como los inconvenientes que presenta el algoritmo EM

- Es un algoritmo estable, porque a medida que incrementamos el número de iteraciones la función de verosimilitud aumenta (excepto cuando llegamos al punto de convergencia) tanto que en la mayoría de las condiciones, podemos afirmar que el algoritmo EM alcance la convergencia, es decir llegar a un máximo local.
- Es un algoritmo fácil de implementar tanto el paso E , que simplemente toma esperanzas sobre distribuciones condicionales de datos completos y el paso M que simplemente requiere estimaciones de verosimilitud de datos completos en cada una de las iteraciones, no requiere mucho espacio de almacenamiento
- El algoritmo EM requiere un alto número de iteraciones comparado con otros procedimientos pero lo compensa por el bajo coste de cada iteración.

Algunas de las desventajas que sufre:

- No tiene un procedimiento incluido para proporcionar una estimación de la matriz de covarianza de las estimaciones de los parámetros. Y puede converger de forma desesperadamente lenta en problemas donde hay demasiada información incompleta.

3.6.8 Conclusiones

El proceso de agrupamiento mediante el algoritmo EM es robusto y flexible, además de ser fácil de implementar y probar, no requiere mucho espacio de almacenamiento, además con el uso de la función de verosimilitud podemos llegar a un resultado óptimo sin llegar a sobreentrenar el algoritmo lo que nos ahorra tiempo de proceso y coste.

Pudiendo aplicarse en infinidad de problemas, como vimos en el apartado de aplicaciones, Entre ellos problemas de clasificación, clustering o agrupamiento, y problemas de aproximación de funciones o regresión.

Podemos afirmar que estamos ante uno de los mejores algoritmos para los problemas de datos incompletos,donde el algoritmo EM muestra su maximo potencial y donde la estimación de máxima verosimilitud resulta difícil debido a la ausencia de alguna parte de los datos.

3.7 Algrotimo SVM

Las máquinas de soporte vectorial o máquinas de vectores de soporte (Support Vector Machines, SVMs) son un conjunto de algoritmos de aprendizaje supervisado desarrollados por Vladimir Vapnik según [VAP95] y su equipo en los laboratorios AT&T. Estos métodos están propiamente relacionados con problemas de clasificación y regresión, Para clasificación el SVM se plantea como un problema de programación lineal en el que buscamos maximizar la distancia entre categorías sujeto a un coste y a un número óptimo de patrones de entrenamiento.

Es un método de clasificación que se basa en encontrar el mejor hiperplano que separa dos conjuntos de datos pertenecientes a dos clases distintas. Para ello, se maximiza la distancia al punto más cercano de cada clase con el fin de obtener el menor error de generalización. Para hallar la frontera de separación, es necesario resolver un problema de optimización usando técnicas de programación cuadrática. A partir del hiperplano ajustado, se pueden clasificar nuevos datos en una de las dos categorías.

Además, la SVM permite la separación de datos no linealmente separables, transformando los datos de entrada a un espacio de mayor dimensión en el que sí pueden ser separados mediante un hiperplano. La transformación se realiza mediante unas funciones denominadas núcleo o kernels. Modificando distintos parámetros de las SVM se pueden obtener diferentes tipos de fronteras de separación. Habitualmente, las SVM se emplean para clasificación binaria.

3.7.1 Introducción

Dado un conjunto de instancias u observaciones que pertenecen a dos clases linealmente separables, el algoritmo SVM busca un plano (o frontera de separación) que separe de la forma más óptima posible las instancias de una clase de otra, a ese plano en ingles se le denomina hyperplane, aquí nos vamos a encontrar con varios planos o fronteras que separen las dos clases, el algoritmo buscara la mejor frontera o plano según un determinado criterio.

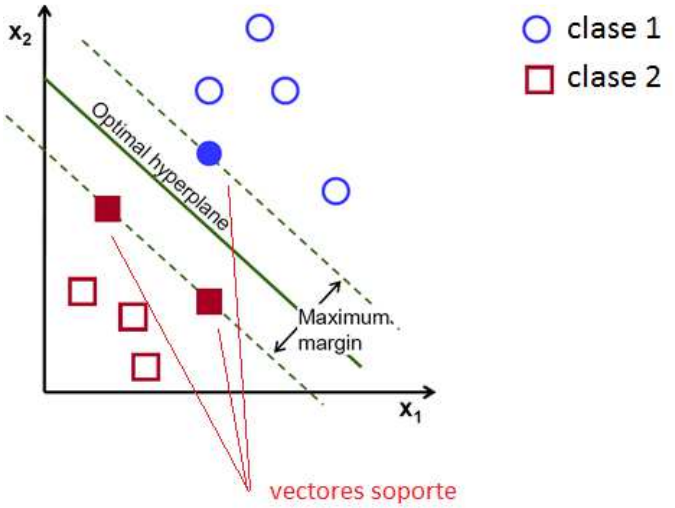


Figura 16 – Vectores soporte

Fuente [opencv3.7.1]

El criterio que toma el algoritmo SVM es buscar un plano que maximiza la distancia o margen que hay entre los puntos más cercanos al plano (vectores soporte) y dicho plano, dicho plano será el plano óptimo que buscara el algoritmo.

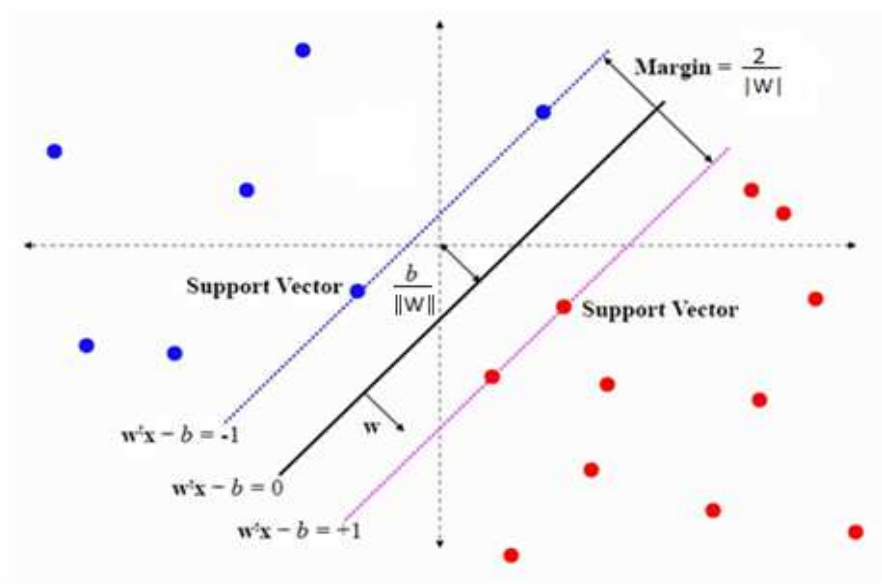


Figura 17 – Plano optimo SVM

Fuente [microsoft3.7.1]

Cada una de esas observaciones consiste en los siguientes datos

Un vector $x_i \in R^n, i = 1,...,\mathrm{l}$

Una etiqueta $y_i \in \{+1,-1\}$

Supongamos que se tiene un hiperplano que separe las muestras positivas (+1) de las muestras negativas (-1), los puntos Xi que están en el plano satisfacen **w . x - b = 0**

Donde

W es el vector normal al hiperplano (el vector de pesos)

$|b|/\|w\|$ Es la distancia perpendicular del hiperplano al origen

$\|w\|$ Es la norma euclídea del vector W

Lo que se quiere es separarlos puntos de acuerdo al valor de Yi en dos hiperplanos diferentes

W . Xi - b >= +1 hiperplano positivo Yi = +1

W . Xi - b <= -1 hiperplano negativo Yi = -1

Simplificando obtenemos $\mathrm{yi}(\mathrm{w}\cdot \mathrm{xi} - \mathrm{b}) \geq +1$

Sea +d la distancia más corta entre el hiperplano positivo y el punto positivo más cercano.

Sea -d la distancia más corta entre el hiperplano negativo y el punto positivo más negativo.

Sea el margen la distancia entre el hiperplano positivo y hiperplano negativo y es igual a $\frac{2}{\|\mathbf{w}\|}$

La idea es encontrar un hiperplano con el máximo margen, lo cual sería un problema de optimización

Maximizar $\frac{2}{\|\mathbf{w}\|}$ sujeto a $\mathrm{yi}(\mathrm{w}\cdot \mathrm{xi} - \mathrm{b}) \geq +1$

También se puede expresar de la siguiente manera

Minimizar $\frac{1}{2}\|\mathbf{w}\|^2$ sujeto a $\mathrm{yi}(\mathrm{w}\cdot \mathrm{xi} - \mathrm{b}) \geq +1$

Usamos multiplicadores de Lagrange (αi) para simplificar el problema

$$L_P \equiv \tfrac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l} \alpha_i \left(y_i \left(\mathbf{w}\cdot\mathbf{x}_i - b\right) - 1\right)$$

Haciendo que las gradientes de Lp respecto a w y b sean cero, se obtienen las siguientes condiciones

$$\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i \qquad\qquad \sum_{i=1}^{l} \alpha_i y_i = 0$$

Remplazando dichas condiciones en Lp obtenemos la siguiente ecuación

$$L_D = \sum_{i=1}^{l} \alpha_i - \tfrac{1}{2} \sum_{i=1,\, j=1}^{l} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

Entonces el problema de optimización pasaría a ser

Maximizar $$L_D = \sum_{i=1}^{l} \alpha_i - \tfrac{1}{2} \sum_{i=1,\, j=1}^{l} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

Sujeto a $$\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i \qquad\qquad \sum_{i=1}^{l} \alpha_i y_i = 0$$

Al resolverlo se obtiene la función del hiperplano

$$\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i$$

$$y = w.x + b$$

$$f(x) = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i.x + b$$

En la ecuación del hiperplano se puede observar que este solo depende de los vectores soporte ya que el resto de puntos cumplen α = 0, esto quiere decir que si se vuelve a calcular el hiperplano solo con los vectores soporte llegaríamos al mismo resultado.

Esta ecuación nos sirve para clasificar un nuevo dato, de tal manera que si se obtiene como resultado de la función f(x), un número negativo, el punto es clasificado como -1 o su clase equivalente, y si se obtiene un número positivo el punto es clasificado como +1 o su clase equivalente.

3.7.2 SVM con Soft-Margin

En el caso ideal un modelo basado en SVM debería producir un hiperplano que separe completamente los datos de las muestras estudias en dos clases, sin embargo en el mundo real no siempre es posible, y si lo es el resultado no se puede generalizar para otros datos.

Esto se conoce como sobreajuste (overfitting).Con el fin de permitir cierta flexibilidad, los SVM manejan un parámetro C que controla la compensación entre errores de entrenamiento y los márgenes rígidos, creando así un margen blando (soft-margin) que permita algunos errores en la clasificación a la vez que los penaliza.

En vez de minimizar $\frac{1}{2}\|\mathbf{w}\|^2$

Buscamos Minimizar $\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \xi_i$

Donde $y_i(\mathrm{w}\cdot x_i - \mathrm{b}) \geq 1 - \xi_i \qquad \xi_i \geq 0$

La variable ξ_i define el error en la clasificación, si ξ_i = 0 entonces no hay error, el parámetro C es el que controla la compensación entre errores y márgenes.

3.7.3 La Función Kernel

En el apartado anterior vimos un conjunto de instancias u observaciones linealmente separables pero en muchas aplicaciones del mundo real no se pueden separar las instancias de forma lineal, la representación por medio del kernel nos ofrece una alternativa a este problema,

Se hace un cambio de espacio mediante una función que transforme los datos de manera que se puedan separar linealmente. Tal función se llama kernel, La función kernel proyecta la información a un espacio de características de mayor dimensión el cual aumenta la capacidad computacional de las máquinas de aprendizaje lineal, en el siguiente grafico vemos como pasamos de un espacio de 2 dimensiones a otro de 3 dimensiones.

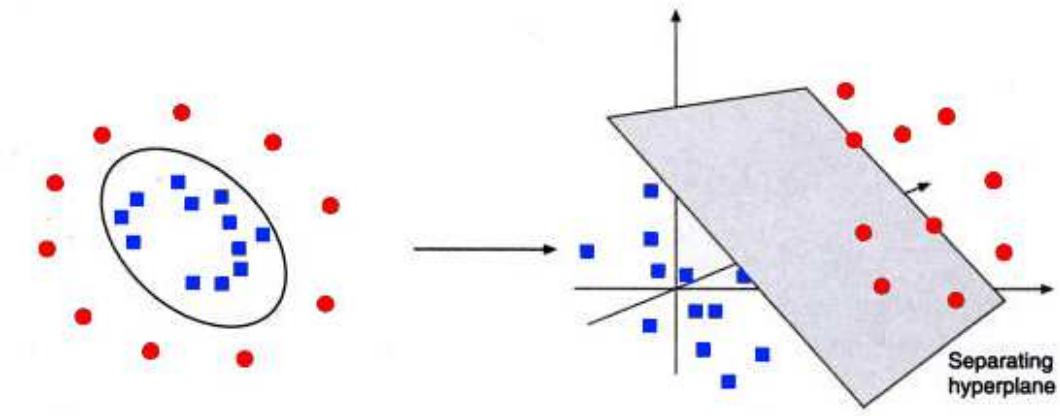


Figura 18 – Función kernel

Fuente [SOM09]

Con lo cual la ecuación para maximizar la distancia o el margen quedaría de la siguiente forma

Maximizar $$L_D = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1, j=1}^{l} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

Sujeto a $\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i \qquad \sum_{i=1}^{l} \alpha_i y_i = 0$

Donde K (Xi, Xj) = Xi . Xj sería nuestra función kernel, Existen varios tipos de funciones kernel lo más destacados son

Kernel Polinomial $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i . \mathbf{x}_j)^n$

Kernel Gausiano radial $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-(\mathbf{x}_i - \mathbf{x}_j)^2 / 2\sigma^2)$

Kernel Perceptrón $K(\mathbf{x}_i, \mathbf{x}_j) = \left\| \mathbf{x}_i - \mathbf{x}_j \right\|$

Kernel Sigmoidal $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\mathbf{x}_i . \mathbf{x}_j - \theta)$

3.7.4 clasificación multiclase

Las SVM se usan habitualmente para problemas de tipo binario. Una de las soluciones para resolver este problema multiclase es convertirlo en varios binarios. Para ello, existen 2 métodos distintos:

- Clasificación 1-v-r (del inglés one-versus-rest): en cada uno de los problemas se considera una clase positiva y las demás negativas, por lo que habrá que hallar tantos hiperplanos como clases existan, este enfoque es el más habitual.

- Clasificación 1-v-1 (del inglés one-versus-one): para cada problema se toman 2 clases de las K totales. Se compara cada clase con cada una de las restantes, lo que supone realizar K (K - 1)/2 clasificaciones.

3.7.5 Aplicaciones

En la última década, una considerable comunidad de teóricos e ingenieros se ha formado alrededor de estos métodos, y se han realizado numerosas aplicaciones prácticas. Hay muchos métodos basados en ellas que aparecen en el estado del arte de diversas tareas de aprendizaje de máquinas. Su fácil uso, su atractivo teórico, y su notable desempeño han hecho de ellas una buena elección para muchos problemas de aprendizaje computacional.

Las Máquinas de Soporte Vectorial han sido aplicadas con éxitos en muchos problemas de la vida real y en diversas áreas: reconocimiento de patrones, Regresión, Multimedia, Bioinformática, inteligencia artificial, categorización de textos y reconocimiento de caracteres escritos hasta la clasificación de datos de expresiones de genes. etc.

Muchas técnicas como árboles de decisión, redes neuronales, algoritmos genéticos, etc., han sido usadas en esas áreas; sin embargo, lo que distingue a las SVM es su sólida fundamentación matemática la cual está basada es la teoría del aprendizaje estadístico.

Además, más que la minimización del error de entrenamiento (riesgo empírico, como es el caso de las redes neuronales), la SVM minimizan el riesgo estructural que expresa una cota superior del error de generalización, es decir la probabilidad de una clasificación errónea sobre ejemplos no vistos todavía.

Esto hace particularmente conveniente a las SVM en muchas aplicaciones con datos dispersos, Ya que ese énfasis especial de las SVM sobre la habilidad de generalización hace de esta aproximación particularmente interesante para aplicaciones del mundo real con limitada cantidad de datos de entrenamiento.

3.7.6 Ventajas e Inconvenientes

Las SVM tienen ciertas ventajas, comparadas con otras técnicas de clasificación:

- Al usar un principio inductivo, que busca la minimización del riesgo estructural, usando una función kernel, le da la una gran capacidad de generalización, incluso es interesante para aplicaciones del mundo real con un conjunto de entrenamiento con una cantidad de datos limitada.
- El proceso de entrenamiento (o aprendizaje) no depende necesariamente del número de atributos, por lo que se comportan muy bien en problemas de alta dimensionalidad. Existen pocos parámetros a ajustar, el modelo solo depende de los datos con mayor información.
- están explícitamente basados en un modelo teórico de aprendizaje más que sobre una analogía relajada con los sistemas de aprendizaje natural u otras heurísticas
- El modelo final puede ser escrito como una combinación de un número muy pequeño de vectores de entrada, llamados vectores de soporte, lo cual se presenta ya que el Lagrangiano de la mayoría de los vectores es igual a cero. El compromiso entre la complejidad del clasificador y el error puede ser controlado Explícitamente.
- tienen garantía teórica sobre su desempeño y tienen un diseño modular que hace posible separar la implementación y el análisis de sus componentes.
- no son afectados por el problema de los mínimos locales, debido a que su entrenamiento se basa en problemas de optimización convexa.

Algunas de sus inconvenientes son:

- En gran medida, la solución al problema, así como su generalización depende del kernel que se use y de los parámetros del mismo, con lo cual se necesita una buena función kernel es decir se necesitan metodologías eficientes para sintonizar los parámetros de inicialización de la SVM.

3.7.7 conclusiones

Las SVM, aplicadas al problema de clasificación, mapean los datos a un espacio de características de mayor dimensión, donde se puede hallar más fácilmente un hiperplano de separación. Este mapeo puede ser llevado a cabo aplicando el kernel, el cual transforma implícitamente el espacio de entrada en un espacio de características de alta dimensión.

El hiperplano de separación es calculado maximizando la distancia de los patrones más cercanos, es decir la maximización del margen. Las SVM pueden ser definidas como un sistema para el entrenamiento eficiente de máquinas de aprendizaje lineal en un espacio de características inducido por un kernel, Las dos características claves de las máquinas de soporte vectorial son:

- La teoría de generalización, la cual conduce a una elección de hipótesis basada en principios teóricos.
- Un kernel de funciones, que introduce no linealidades en el espacio de hipótesis sin requerir explícitamente algoritmos no lineales.

Aunque existen muchas técnicas alternativas para enfrentar problemas de regresión y clasificación, las máquinas de soporte vectorial han sido desarrolladas como una herramienta robusta para regresión y clasificación en dominios complejos y ruidosos. Las SVM pueden ser usadas para extraer información relevante a partir de conjunto de datos y construir algoritmos de clasificación o de regresión rápidos para datos masivos.

Capítulo 4

4. Arquitectura y Diseño

4.1 Identificación del entorno tecnológico

El propósito de identificar el entorno tecnológico es el de ayudar al desarrollo del proyecto siguiendo ciertos estándares de calidad que define la ingeniería de software y que nos ayudaran al éxito en el desarrollo del sistema.

En nuestro caso el sistema a desarrollar es una Aplicación Web por lo que va a poseer una serie de características que van a delimitar previamente ciertos parámetros de su entorno de desarrollo, de pruebas y puesta en producción.

El diseño técnico y posterior desarrollo de esta aplicación se hará en base a un Arquitectura J2EE, la implantación de esta arquitectura tiene como objetivo mejorar la eficiencia del desarrollo, ejecución y mantenimiento de la aplicación. Una arquitectura J2EE correctamente construida provee de las siguientes ventajas:

- Alta reutilización de componentes
- Fácil mantenimiento
- Escalabilidad
- Alto rendimiento
- Alta disponibilidad
- Alta fiabilidad
- Estandarización
- Fácil monitorización
- Alta seguridad

Una característica esencial del entorno tecnológico que debemos seguir es realizar un diseño Orientado a Objetos (en adelante OO), la implantación de la arquitectura J2EE debe estar enfocada en todo momento desde un punto de vista OO, este diseño OO debe ser eficiente y estar bien estudiado, analizado e implementado para así garantizar el éxito de la implementación de la aplicación a través del uso de una arquitectura como lo es J2EE.

4.2 Arquitectura

Dentro de la arquitectura J2EE existen distintos modelos de desarrollo, el modelo elegido para esta aplicación será el comúnmente conocido como 'modelo 2', este modelo está basado en el paradigma del patrón de diseño Model-View-Controller (MVC), que pretende separar una interfaz lógica de usuario en sus componentes lógicos. Se consigue de esta manera la reutilización de las unidades lógicas y evitar que los cambios en la interfaz afecten a dichas unidades. En realidad el patrón MVC es más que un simple patrón. Es una separación de responsabilidades común en el diseño de cualquier aplicación:

- Modelo (Model): Hace referencia a la representación en el mundo real de un dominio.
- Vista (View): Hace referencia a las formas de representar los datos que se quieren manejar.

- Controlador (Controller): Hace referencia a las acciones concretas que puede realizar el sistema.

La arquitectura de modelo 2 trabaja de la siguiente forma:

- La solicitud llega al controlador.
- El controlador realiza una acción con los parámetros proporcionados.
- El controlador pasa el control a la vista para obtener la respuesta.
- La vista toma como referencia el modelo de dominio para construir la representación.
- La vista devuelve la respuesta al usuario.

Ventajas del Modelo 2 – J2EE

- **Flexibilidad:** Se separa la aplicación en sus componentes según la importancia de lo que hacen. Esto permite emplear dichos componentes en nuevas vistas o acciones sin necesidad de reescribir le código cada vez. Podemos incluso reutilizar componentes en otras plataformas de aplicación
- **Reutilización:** La división en componentes permite reutilizar un marco para proporcionar gran parte de la cohesión que mantiene unida la aplicación.
- **Seguridad:** Manejando todas las opciones desde un controlador central, podemos configurar y gestionar fácilmente el control de acceso a nuestros datos y acciones.

Debido a la utilización de la arquitectura J2EE de Modelo 2 el sistema estará dividido en las siguientes capas:

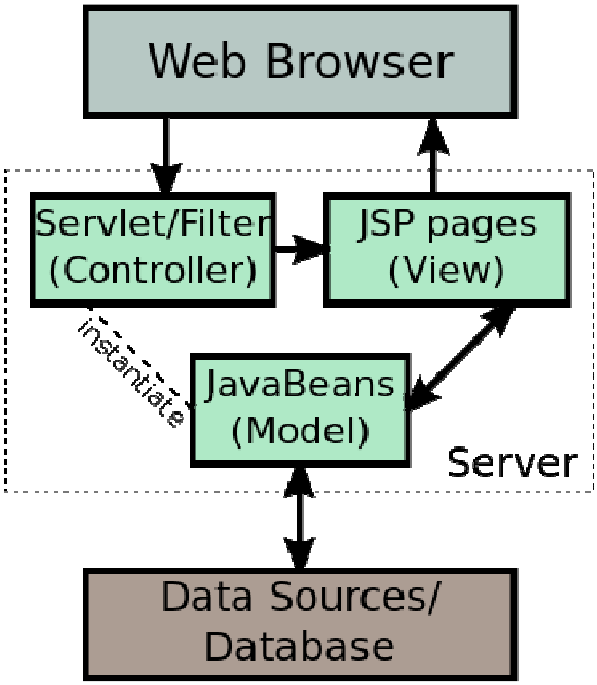


Figura 19 – J2EE Modelo 2

Fuente [wikipedia4.2]

Para abordar la descripción de cada una de estas capas es necesario ahondar previamente en la arquitectura del sistema y de los Frameworks que se han utilizado para simplificar y estandarizar el desarrollo de la aplicación.

4.3. Descripción de los Frameworks utilizados

Entendemos Framework como un conjunto de utilidades, una representación de una arquitectura del software (bien sea en forma de API librería JavaScript o lo que se quiera) que encapsula una estructura y manera de trabajo, un comportamiento horizontal, que es usada por las aplicaciones del dominio. Así pues representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y "manera de trabajo" la cual extiende o utilizan las aplicaciones del dominio.

En la Aplicación se van a utilizar los siguientes Frameworks:

- SPRING: encapsula el patrón de diseño MVC

- Dojo: Framework JavaScript y AJAX

4.3.1- SPRING Framework

Spring Framework es una plataforma que nos proporciona una infraestructura que actúa de soporte para desarrollar aplicaciones Java. Spring maneja toda la infraestructura y así podemos centrarnos en la aplicación. Diciéndolo más coloquialmente, Spring es el "pegamento" que une todos los componentes de la aplicación, maneja su ciclo de vida y la interacción entre ellos.

Spring Framework es un contenedor ligero ("lightweight container") En el caso de una aplicación Web, basta con un contenedor de servlets como Tomcat o Jetty. Pero Spring no solo se puede usar para crear aplicaciones Web, se podría usar para cualquier aplicación java, aunque su uso habitual sea en entornos Web, nada te impide utilizarlo para cualquier tipo de aplicación.

Si bien las características fundamentales de Spring Framework pueden ser usadas en cualquier aplicación desarrollada en Java, existen variadas extensiones para la construcción de aplicaciones Web sobre la plataforma Java EE. A pesar que no impone ningún modelo de programación en particular, este Framework se ha vuelto popular en la comunidad al ser considerado una alternativa, sustituto, e incluso un complemento al modelo EJB (Enterprise JavaBean).

En general, estas son algunas de las características de Spring:

- Simplificación de la programación orientada a aspectos.
- Simplificación del acceso a datos.
- Simplificación e integración con JEE
- Soporte para planificación de trabajos.
- Soporte para envió de mail.
- Interacción con lenguajes dinámicos (como BeanShell, JRuby, y Groovy).
- Soporte para acceso a componentes remotos.
- Manejo de Transacciones.
- Su propio Framework MVC.
- Su propio Web Flow.
- Manejo simplificado de excepciones.

Spring Framework comprende diversos módulos que proveen un rango de servicios:

- **Contenedor de inversión de control**: permite la configuración de los componentes de aplicación y la administración del ciclo de vida de los objetos Java, se lleva a cabo principalmente a través de la inyección de dependencias.
- **Programación orientada a aspectos**: habilita la implementación de rutinas transversales.
- **Acceso a datos**: se trabaja con RDBMS en la plataforma java, usando Java Database Connectivity y herramientas de Mapeo objeto relacional con bases de datos NoSQL.
- **Gestión de transacciones**: unifica distintas APIs de gestión y coordina las transacciones para los objetos Java.
- **Modelo vista controlador**: Un Framework basado en HTTP y servlets, que provee herramientas para la extensión y personalización de aplicaciones Web y servicios Web REST.
- **Framework de acceso remoto**: Permite la importación y exportación estilo RPC, de objetos Java a través de redes que soporten RMI, CORBA y protocolos basados en HTTP incluyendo servicios Web (SOAP).
- **Convención sobre configuración**: el módulo Spring Roo ofrece una solución rápida para el desarrollo de aplicaciones basadas en Spring Framework, privilegiando la simplicidad sin perder flexibilidad.
- **Procesamiento por lotes:** Básicamente se trata de operaciones de negocio que necesitan procesar de forma automática grandes volúmenes de información sin la interacción del usuario para ganar en eficiencia, Spring provee un modulo para el procesamiento de alto volumen con funciones reutilizables, incluyendo Registro / trazado, gestión de transacciones, estadísticas de procesamiento de trabajo, y gestión de recursos.
- **Autenticación and Autorización**: procesos de seguridad configurables que soportan un rango de estándares, protocolos, herramientas y prácticas a través del sub-proyecto Spring Security (formalmente Acegi Security System for Spring).
- **Administración Remota:** Configuración de visibilidad y gestión de objetos Java para la configuración local o remota vía JMX.
- **Mensajes**: Registro configurable de objetos receptores de mensajes, para el consumo transparente desde la a través de JMS, una mejora del envío de mensajes sobre las API JMS estándar.
- **Testing**: Soporte de clases para desarrollo de unidades de prueba e integración.

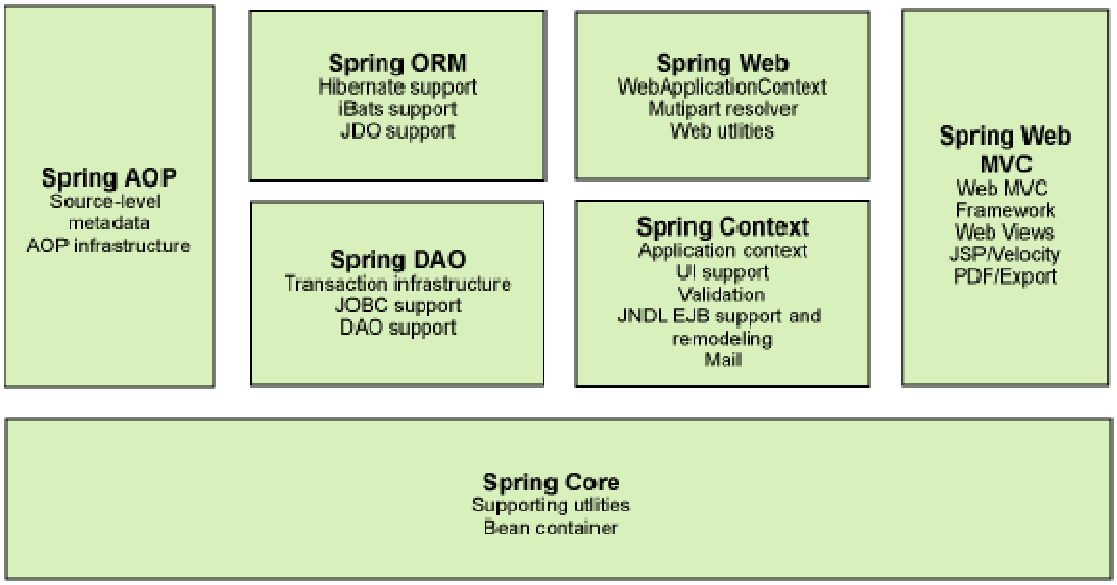


Figura 20 – Módulos Spring

Fuente [Spring4.3.1]

La versión más reciente de Spring es La versión 3 ,es una versión revisada y mejorada de la versión estable anterior (2.5), que incluye nuevas características, entre las que se incluyen:

- Soporte para Java 5: Proporciona configuración basada en anotaciones y soporta características como varargs y generics, además la parte Web es compatible con las versiones 1.4 y 5 de Java EE. Debido a esta nueva característica, ahora es necesario tener el JRE versión 5 o superior.
- Lenguaje de Expresiones (SpEL): En esta nueva versión se incluye un lenguaje de expresiones que puede ser usando cuando se definen beans, tanto en XML como con anotaciones y también da soporte a través de todos los módulos de Spring.
- Soporte para Servicios Web REST: Ahora Spring soporte servicios web de tipo REST.
- Soporte para Java EE6: Ofrece soporte de características como JPA 2.0, JSF 2.0 y JRS 303 (validaciones de Beans).
- Soporte para bases de datos embebidas: Un soporte conveniente para bases de datos embebidas como HSQL, H2 y Derby.
- Soporte para formateo de datos mediante anotaciones: Ahora los campos de fecha, divisas, etc., serán formateados automáticamente y convertidos usando anotaciones.

- Nueva organización de los módulos: Los módulos han sido revisados y separados en diferentes paquetes, mas organizados, de acuerdo a su funcionalidad, estos son los nuevos paquetes:
    - org.springframework.aop
    - org.springframework.beans
    - org.springframework.context
    - org.springframework.context.support
    - org.springframework.expression
    - org.springframework.instrument
    - org.springframework.jdbc
    - org.springframework.jms
    - org.springframework.orm
    - org.springframework.oxm
    - org.springframework.test
    - org.springframework.transaction
    - org.springframework.web
    - org.springframework.web.portlet
    - org.springframework.web.servlet
    - org.springframework.web.struts

En nuestra plataforma nos interesa el modulo de Spring MVC encargado del patrón Modelo-vista-controlador, es un marco de trabajo muy popular a la hora de construir aplicaciones Web basadas en Java. Entre sus características clave se incluyen:

⇨ Una arquitectura general basada en los principios de diseño de Modelo-Vista-Controlador (MVC) en el que todas las peticiones son procesadas por el controlador que realiza el control (valga la redundancia) de todas la peticiones y despacha las peticiones a los componentes de la aplicación apropiados, basándose en los identificadores lógicos que reducen el acoplamiento entre capas.

- Capacidades de manejo de formularios, y un marco de trabajo de validación que externaliza la configuración de un conjunto de chequeos de exactitud que se aplican a los valores de los campos de entrada, además implementa estos chequeos tanto en el lado del cliente como en el lado del servidor.
- Un conjunto de etiquetas JSP personalizadas que simplifican el proceso de crear las etiquetas HTML de la aplicación para la capa "Vista", y que trabaja de forma integrada con las capacidades de manejo de formularios y la arquitectura general del controlador.

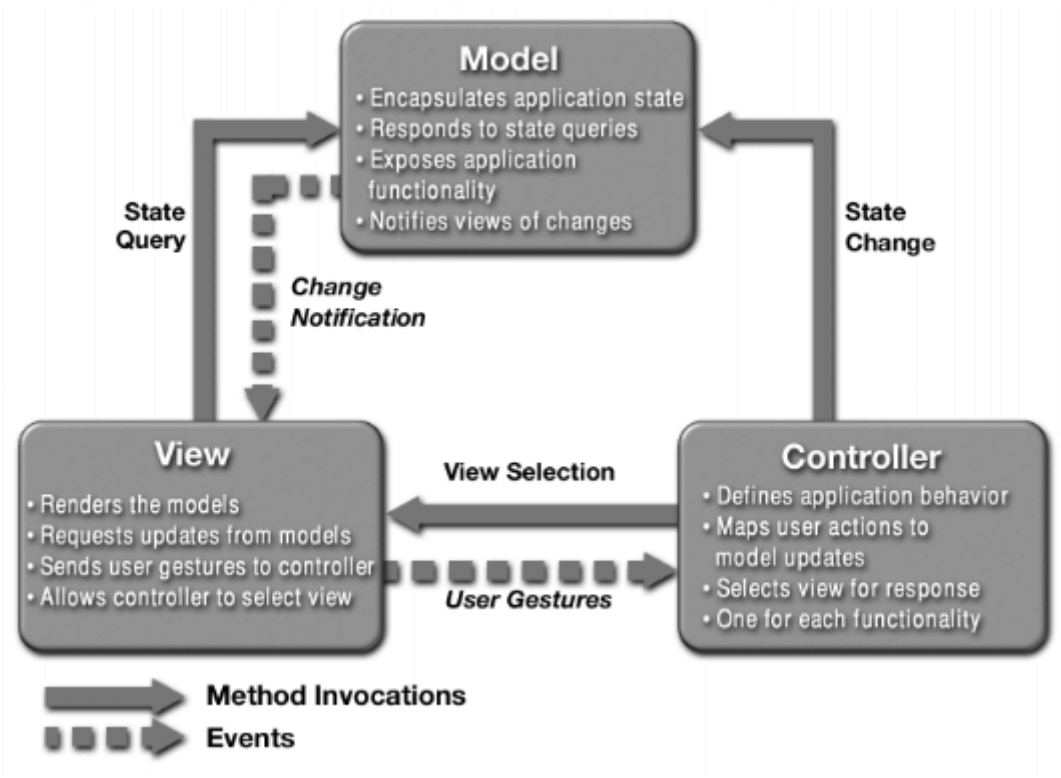


Figura 21 – Modelo Vista Controlador

Basado en fuente [netbeans4.3.1]

Esta combinación de características sumadas a la madurez de la implementación y a la importante comunidad que lo respalda (documentación, libros, artículos, foros de soporte, etc.) son algunas de las razones por las que este Framework se ha convertido en una arquitectura 'de facto' para los desarrolladores J2EE.

4.3.2. DOJO TOOLKIT

Dojo es un Framework que contiene APIs y widgets (controles) para facilitar el desarrollo de aplicaciones Web que utilicen tecnología AJAX. Contiene un sistema de empaquetado inteligente, los efectos de UI, drag and drop APIs, widget APIs, abstracción de eventos, almacenamiento de APIs en el cliente, e interacción de APIs con AJAX.

Resuelve asuntos de usabilidad comunes como pueden ser la navegación y detección del navegador, soportar cambios de URL en la barra de Urls para luego regresar a ellas (bookmarking), y la habilidad de degradar cuando AJAX/JavaScript no es completamente soportado en el cliente. Es conocido como "la navaja suiza del ejército de las bibliotecas JavaScript". Proporciona una gama más amplia de opciones en una sola biblioteca JavaScript y es compatible con navegadores antiguos.

Una característica importante de las aplicaciones AJAX es la comunicación asíncrona entre el navegador y el servidor. Tradicionalmente, se realizaba con el comando JavaScript XMLHttpRequest.

En una aplicación AJAX, un evento de usuario dispara la ejecución de un motor AJAX, construido con JavaScript el cual, usando el objeto XmlHTTPRequest establecerá una conexión asíncrona con el servidor, solicitando la ejecución de algún mecanismo similar a un procedimiento remoto (puede ser un servicio Web, un action de struts o simplemente un Servet). Este procedimiento nos devolverá, un documento XML, que será procesado por el API DOM y actualizará una parte de la página que se está visualizando mientras que el usuario puede seguir trabajando en otra parte de la misma.

Dojo provee de una capa de abstracción (dojo.io.bind) para varios navegadores Web con la que se pueden usar otros transportes (como IFrames ocultos) y diferentes formatos de datos.

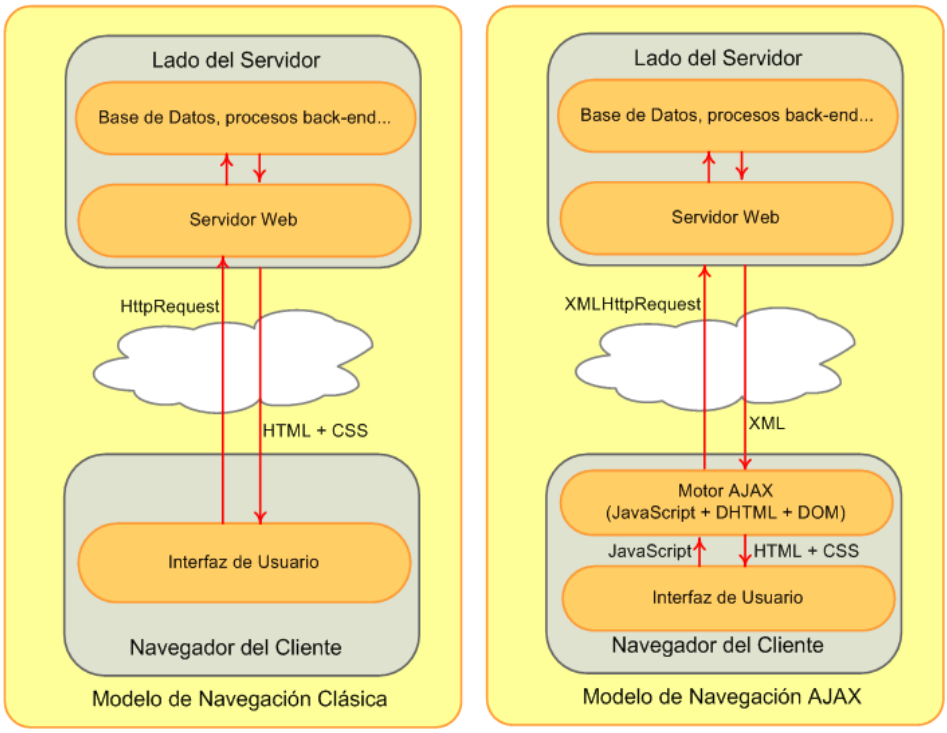


Figura 22 – Navegación AJAX

Dojo también aporta a nuestra plataforma la capacidad de dibujar gráficos para representar los resultados de datos después de aplicar los algoritmos aprendizaje automático.

Complementos

Los complementos de Dojo son componentes preempaquetados de código JavaScript, HTML y CSS que pueden ser usados para enriquecer aplicaciones Web.

- Menús, pestañas y tooltips.
- Tablas, gráficos dinámicos y dibujados de vectores 2D.
- Efectos de animación y la posibilidad de crear animaciones personalizables.
- Soporte para arrastrar y soltar.
- Formularios y rutinas de validación para los parámetros.
- Calendario, selector de tiempo y reloj.
- Editor online de texto enriquecido.
- Núcleo de componentes (dijit) accesible desde versiones anteriores y lector de pantalla.

4.4. Descripción de los componentes del sistema

Los componentes del proyecto estarán definidos y divididos en las tres grandes capas dadas por el patrón MVC: Presentación, Control y Negocio.

Las capas de presentación y control estarán gestionadas a través del Framework Spring, más concretamente el modelo SpringMVC, la capa de presentación estará formada en su mayoría por paginas jsp. El uso de Spring como marco para el desarrollo de nuestro MVC nos da implementado la parte más compleja del Controller: el Servet controlador central que va a redirigir el flujo de la aplicación dependiendo de las peticiones del usuario y basándose en ficheros de configuración XML y las debidas anotaciones que se incluyen en los objetos controladores.

El modelo contendrá por un lado las clases correspondientes que se van a encargar de gestionar el flujo de datos con la capa de presentación y la Lógica de negocio que estará formada por clases java que gestionarán la lógica del negocio.

Estas clases java serán las encargadas de realizar las funcionalidades correspondientes a la modificación y manipulación de los datos de la aplicación en lo referente a la lógica del negocio y de la persistencia de los datos. Resumiendo, en el modelo de negocio de la aplicación nos vamos a encontrar los siguientes tipos de objetos:

- **Objetos Controller**: clases que van a ser llamados por el Servet controlador principal para ejecutar las distintas peticiones del navegador.
- **Objetos Validator**: se encargan de validar los datos de la interfaz de usuario a través de Spring Framework.
- **Objetos BO**: Los “Business Object” encargados de gestionar el acceso a datos y la aplicación de algoritmos sobre esos datos.
- **Objetos Bean**: Los “JAVA Bean”, Se usan para encapsular varios objetos en un único objeto para hacer uso de un solo objeto en lugar de varios más simples.

Es importante que la capa de modelo esté a su vez divida en dos partes, la parte control de flujo de Spring y los objetos de la lógica de negocio. Su razón de ser es la de evitar el acoplamiento, es decir, que unas capas del programa que hemos desarrollado no interfieren directamente en el funcionamiento de otras y por tanto puedan ser separadas en un momento dado. Por ejemplo podemos usar todo el modelo de negocio sin necesidad de realizar cambios en una aplicación Web o

de escritorio sólo modificando el interfaz de usuario. Podemos usar todo el front de una aplicación Web en dos entornos similares simplemente 'alimentando' con una nueva capa de negocio los objetos de la presentación.

Por otro lado las tecnologías J2EE progresan a gran velocidad, todos los meses nos encontramos ante la aparición de nuevas tecnologías y estándares que hacen la aplicación java más robustas y de mayor calidad para el usuario final (concepto de 'experiencia de usuario'). Aunque una vez definida la arquitectura básica de una aplicación es muy difícil que esta cambie a veces es inevitable que esto suceda por quedar una aplicación muy por debajo de los estándares de calidad establecidos, tener un software con escaso acoplamiento, significa que podemos 'rescatar' gran parte del código de una aplicación que todavía es válido o que podemos incluir pequeñas mejoras sin necesidad de grandes cambios en la aplicación.

Otro punto importante en la determinación del entorno tecnológico es la especificación del servidor de aplicaciones que se va a usar, en este caso usaremos Apache Tomcat 7 que será el servidor elegido para el despliegue de nuestra aplicación Web.

CUADRO RESUMEN DE LOS COMPONENTES DE SISTEMA	
SERVIDOR DE APLICACIONES	Tomcat 7
Arquitectura	J2EE - SPRING
ENTORNO DE DESARROLLO	Eclipse Kepler
LENGUAJES DE DESARROLLO	Java, JavaScript, XML, HTML
PÁGINAS DE PRESENTACIÓN	JSP, JSTL, HTML
FRAMEWORK PRESENTACIÓN (MVC)	Spring MVC
Lenguaje de modelado de sistemas de software	UML
FRAMEWORK JavaScript	Dojo Toolkit
Gráficos JavaScript	Dojo Toolkit
Estilos Web	CSS

Tabla 12 – Cuadro resumen de los componentes del sistema

4.5 Estructura de clases

La estructura principal de paquetes para la aplicación Web de aprendizaje automático será ***com.machine.learning,*** desde esta raíz las clases se agruparán de la siguiente forma:

- ***com.machine.learning.bean*** :

  Objetos Java Beans.

- ***com.machine.learning.bo:***

  Objetos de la lógica de negocio.

- ***com.machine.learning.controller***: Objetos controlador .

- ***com.machine.learning.connectAPI:*** Objetos para conectar nuestra aplicación con una API de algoritmos de aprendizaje automático.

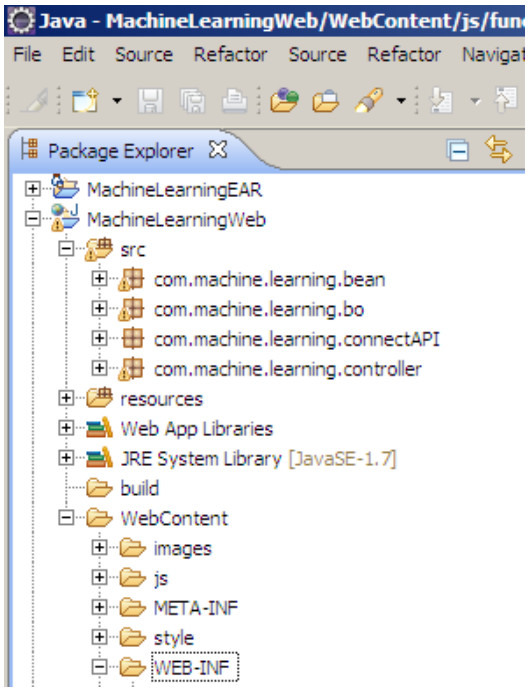


Figura 23 – Estructura de clases

A continuación se explica detalladamente el contenido de cada uno de los paquetes de la aplicación.

4.5.1 Objetos de la lógica de negocio

El paquete ***com.machine.learning.bo*** se encarga de la lógica de negocio de la aplicación, contiene las siguientes clases:

ProcesadorDatos.java: clase encargada de gestionar la entrada de datos a la aplicación y su posterior transformación en Java Beans, creando un conjunto de datos en forma de objeto Java Beans y sus correspondientes instancias y atributos y todas sus propiedades.

EntrenarAlgoritmos.java: clase encargada de entrenar algoritmos de aprendizaje automático con un conjunto de

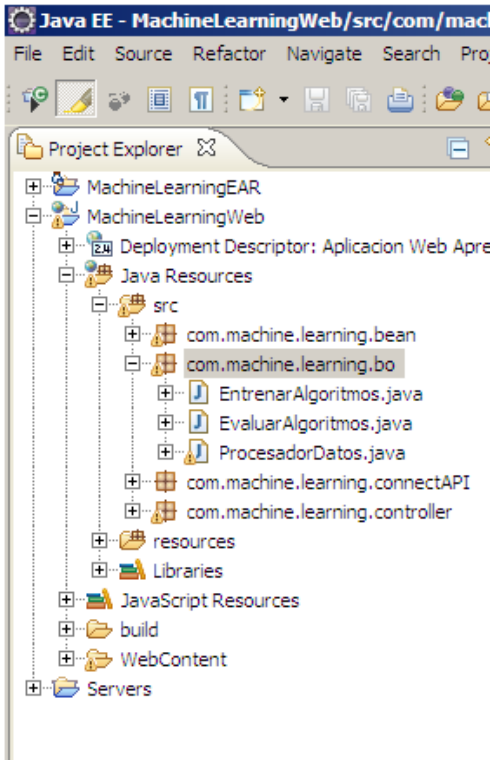


Figura 24

Objetos de la lógica de negocio

datos dado, lo hace a través de llamadas al connectAPI, esta clase no necesita saber qué tipo de API estamos usando para entrenar algoritmos, almacena el resultado en los Java Beans, es llamada por los objetos controladores.

EvaluarAlgoritmos.java: clase encargada de evaluar algoritmos de aprendizaje automático con un conjunto de datos dado, lo hace a través de llamadas al connectAPI, esta clase no necesita saber qué tipo de API estamos usando para evaluar los algoritmos, almacena el resultado en los Java Beans, es llamada por los objetos controladores.

4.5.2 Objetos Java Beans

El paquete ***com.machine.learning.bean*** contiene los objetos Java Beans que se encargan de almacenar el resultado tanto del entrenamiento como la evaluación de algoritmos además de almacenar el conjunto de datos a entrenar y almacena todo lo anterior en objetos java Beans, contiene las siguientes clases:

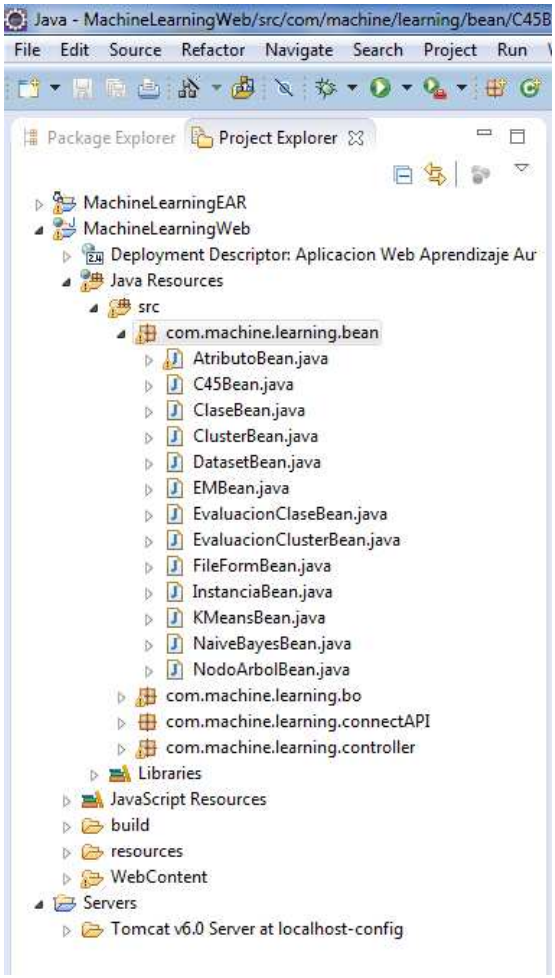


Figura 25 – Objetos Java Beans

AtributoBean.java: los objetos de esta clase almacenan la información que corresponde a un atributo, guarda valores como nombre del atributo, tipo, posibles valores que puede tomar, etiquetas , máximo , mínimo , desviación estándar, media y valor.

ClaseBean.java: los objetos de esta clase almacenan la información que corresponde a una clase, guarda valores como probabilidad, índice, numero de instancias, y un listado de atributos, este objeto se usa en técnicas de clasificación en nuestro caso se usa para almacenar el resultado de entrenamiento y evaluación de los algoritmos C4.5 y Naive Bayes.

ClusterBean.java: los objetos de esta clase almacenan la información que corresponde a un cluster, guarda valores como probabilidad, índice, numero de instancias, y un listado de atributos, este objeto se usa en técnicas de clustering nuestro caso se usa para almacenar el resultado de entrenamiento de los algoritmos EM y KMeans.

DatasetBean.java: los objetos de esta clase almacenan la información que corresponde a un conjunto de datos, guarda valores como nombre del conjunto de datos, número de atributos, numero de instancias, una lista de instancias y una lista de atributos.

EMBean.java: los objetos de esta clase almacenan la información que corresponde al resultado del entrenamiento del algoritmo de clustering esperanza-maximización o algoritmo EM, guarda valores como nombre del algoritmo, numero de iteraciones, número máximo de iteraciones, numero de clusteres, Un Array del número de instancias que hay en cada cluster, el loglikelihood y un conjunto de datos almacenado en un Java Bean.

EvaluacionClaseBean.java: los objetos de esta clase almacenan la información que corresponde al resultado de la evaluación de algoritmos que usen técnicas de clasificación en nuestro caso C4.5 y Naive Bayes , guarda valores como nombre del algoritmo, coeficiente kappa, la precisión del algoritmo ,numero instancias clasificadas correctamente, numero de instancias clasificadas incorrectamente ,porcentaje de instancias clasificadas correctamente, porcentaje de instancias clasificadas incorrectamente, numero de instancias total, numero de atributos ,matriz de confusión y el nombre y etiquetas del atributo que se usa como clase discriminatoria .

EvaluacionClusterBean.java: los objetos de esta clase almacenan la información que corresponde al resultado de la evaluación de algoritmos que usen técnicas de clustering en nuestro caso EM y KMeans, guarda valores como numero de iteraciones, suma de errores, loglikelihood, numero de clusteres, numero de instancias en cada cluster, porcentaje de instancias que hay en cada cluster y el numero de instancias.

InstanciaBean.java: los objetos de esta clase almacenan la información que corresponde a una instancia o muestra del conjunto de datos a entrenar, guarda valores como número cluster o clase a la que pertenece la instancia y un hashtable en el cual cada ocurrencia tiene la etiqueta de un atributo y su correspondiente valor.

C45Bean.java: los objetos de esta clase almacenan la información que corresponde al resultado del entrenamiento del algoritmo de clasificación C4.5, guarda valores como numero de clases, un ArrayList de Java Beans que contienen clases, numero de instancias, un Java Bean que contiene el conjunto de datos a entrenar, el nombre del algoritmo, el nombre de la clase discriminatoria, un Java Bean que contiene los nodos del árbol resultado del entrenamiento y el número de elementos que hay en dicho árbol.

KMeansBean.java: los objetos de esta clase almacenan la información que corresponde al resultado del entrenamiento del algoritmo de clustering KMeans, guarda valores como numero de clusteres, un ArrayList para almacenar los centroides que obtenemos como resultado, un Java Bean que contiene el conjunto de datos a entrenar, el nombre del algoritmo, la suma de errores cuadráticos, un Array que contiene el numero de instancias que pertenecen a cada cluster.

NaiveBayesBean.java: los objetos de esta clase almacenan la información que corresponde al resultado del entrenamiento del algoritmo de clasificación Naive Bayes, guarda valores como numero de clases, un ArrayList de java beans que contienen las clases, numero de instancias, un java bean que contiene el conjunto de datos a entrenar, el nombre del algoritmo, el nombre de la clase discriminatoria.

NodoArbolBean.java: los objetos de esta clase almacenan la información que corresponde al nodo de un árbol, guarda valores como la etiqueta del nodo, referencia del nodo, referencia del nodo padre, y una lista de Java Beans que contiene nodos hijos.

4.5.3 Objetos Controladores

El paquete ***com.machine.learning.controller*** contiene los controladores que van a ser llamados por el Servlet controlador principal para ejecutar las distintas peticiones del navegador, este paquete junto a los ficheros de configuración XML de Spring corresponde al modulo controlador.

Se encargan de recoger datos del usuario y hacer una petición y ejecutar las acciones necesarias de la lógica de negocio y devolver el resultado al modulo de vista, contiene las siguientes clases:

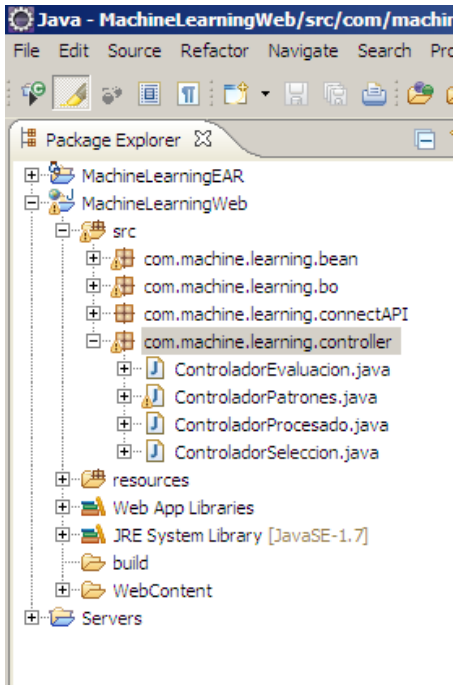


Figura 26 - Objetos Controladores

Controlador Selección

Esta clase se encarga de controlar y gestionar las peticiones de usuarios correspondientes a la carga de un conjunto de datos a entrenar en la aplicación, llama a los correspondientes

objetos de la lógica de negocio para transformar esos datos recogidos desde un fichero de texto y convertirlos en sus correspondientes Java Beans, carga el resultado obtenido desde las clases de lógica de negocio en la pagina jsp *seleccion.jsp.*

Controlador Procesado

Esta clase se encarga de controlar y gestionar las peticiones de usuarios para obtener las propiedades de un determinado atributo, como requisito necesita un conjunto de datos con sus instancias y atributos cargado en la aplicación en forma de Java Beans, carga el resultado obtenido desde las clases de lógica de negocio en la pagina jsp *preprocesado.jsp.*

Controlador Patrones

Esta clase se encarga de controlar y gestionar las peticiones de usuarios que corresponden al entrenamiento de algoritmos sean de clustering o de clasificación como requisito necesita un conjunto de datos con sus instancias y atributos cargado en la aplicación en forma de Java Beans, carga el resultado obtenido desde las clases de lógica de negocio en la pagina jsp *patrones.jsp.*

Controlador Evaluación

Esta clase se encarga de controlar y gestionar las peticiones de usuarios que corresponden a la evaluación de algoritmos sean de clustering o de clasificación, como requisito necesita un conjunto de datos con sus instancias y atributos cargado en la aplicación en forma de Java Beans, carga el resultado obtenido desde las clases de lógica de negocio en la pagina jsp *evaluacion.jsp.*

4.5.4 Objetos connectAPI

En nuestra aplicación los algoritmos de aprendizaje automático no están implementados, para ello utilizamos una API de aprendizaje automático que nos proporciona la implementación de dichos algoritmos, en nuestro caso es la API que proporciona WEKA.

Para hacer nuestra aplicación independiente de dicha API y con tal de que se pueda usar otra API distinta que proporcione la implementación de los algoritmos usados.

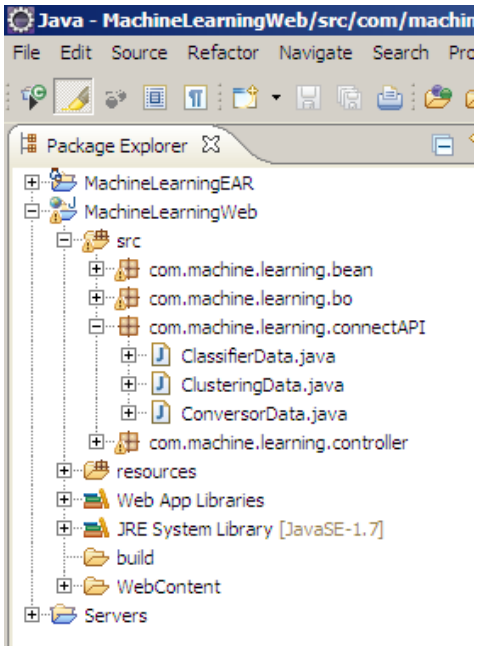


Figura 27 - Objetos connectAPI

Se ha implementado un modulo de conexión o separación que hace que nuestra aplicación no depende de la API a utilizar.

El paquete ***com.machine.learning.connectAPI*** contienen las clases que actúan como una capa de separación entre la API usada y la aplicación, se encarga tanto de la conversión de datos y Java Beans como la clasificación y clustering de datos, solo los objetos de las clases de lógica de negocio invocan estas clases para entrenar y/o evaluar algoritmos y lo hacen sin necesidad de saber que API de algoritmos se está usando, de tal forma que si queremos cambiar de API solo tendríamos que implementar las clases de este modulo según las especificaciones de la API nueva, sin necesidad de cambiar nada en la aplicación.

Se puede encapsular este modulo en un archivo JAR, y para cada API de implementación de algoritmos tendríamos su JAR correspondiente, de tal forma que solo cambiando el JAR tendríamos acceso a otra API, este paquete contiene las siguientes clases:

ClassifierData.java

Esta clase se encarga de entrenar y evaluar algoritmos de clasificación según las especificaciones de la API de implementación de algoritmos, cambiando la implementación de esta clase se puede usar una API u otra, los métodos de esta clase solo son invocados desde las clases de lógica de negocio.

ClusteringData.java

Esta clase se encarga de entrenar y evaluar algoritmos de clustering según las especificaciones de la API de implementación de algoritmos, cambiando la implementación de esta clase se puede usar una API u otra, los métodos de esta clase solo son invocados desde las clases de lógica de negocio.

ConversorData.java

Esta clase se encarga de hacer las conversiones oportunas entre Java Beans de nuestra aplicación y Java Beans de la API de implementación de algoritmos, cambiando la implementación de esta clase se puede usar una API u otra, los métodos de esta clase solo pueden ser llamados por clases del paquete ***com.machine.learning.connectAPI.***

4.6 Estructura de la interfaz

La estructura de la interfaz de usuario para la aplicación Web estará en la carpeta WebContent que actuara como raíz y a partir de la cual los elementos se agruparan de la siguiente manera:

- ***Carpeta Images*** : contiene las imágenes.
- ***Carpeta js :*** contiene los ficheros javascript.
- ***Carpeta style*** : contiene los ficheros de hojas de estilos CSS.
- ***Carpeta META-INF***
- ***Index.jsp*** *: es la página de inicio de la aplicación.*
- ***Carpeta WEB-INF****:* Contiene toda la información de configuración necesarias para la aplicación Web, además de los ficheros JSP y librerías externas.

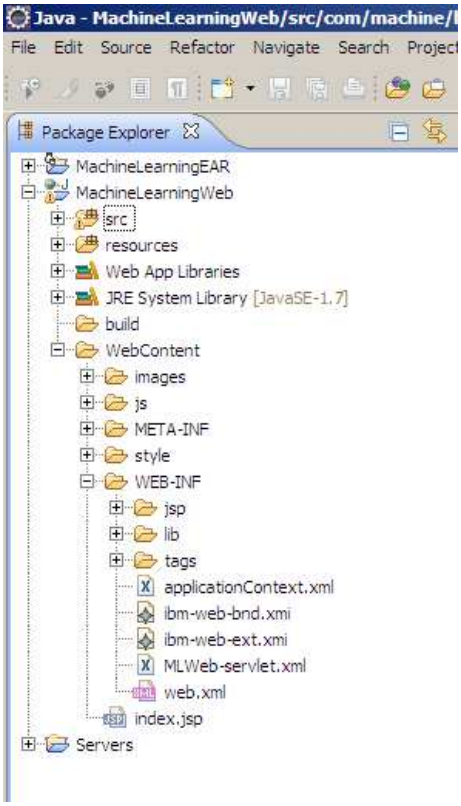


Figura 28 - Estructura de la interfaz

4.6.1 Archivos de configuración XML

Antes de empezar a describir cada uno de los ficheros de configuración de la aplicación explicamos elementos básicos para la configuración de Spring en la aplicación.

ServletContext o ApplicationContext

El ServletContext es el lugar donde habitan los beans (recursos instanciados y configurados) de Spring, solo existe un ServletContext por aplicación. Por otra parte, cada Servlet tiene su propio ServletConfig.

DispatcherServlet

Es un handler de Spring y su tarea es procesar las llamadas realizadas a la aplicación y determinar que Controlador será el encargado de atenderla para su resolución, Desde la versión 3 de Spring, se utilizan las Java Annotations, para configurar los recursos de Spring, en nuestro caso particular hemos usado esa técnica para configurar los controladores de la aplicación, Java Annotations, como otras muchas características del proyecto, dependen de DispatcherServlet.

ContextLoaderListener

Es un listener de Spring que se encarga de realizar la configuración automáticamente en tiempo de despliegue del ApplicationContext (o ServletContext) de la aplicación, y Crea un WebApplicationContext que nos facilita el acceso a ApplicationContext a través del método getServletContext().

La aplicación contiene los siguientes ficheros XML de configuración:

- ***applicationContext.xml***

  Fichero de configuración del contexto de una aplicación en Spring (AplicationContext o ServletContext), en el se declaran los beans de nuestra aplicación Spring, el ContextLoaderListener busca en este documento la declaración de beans para desplegarlos en el root ServletContext (padre) de la aplicación.

- ***MLWeb-servlet.xml***

  A través de este fichero xml configuramos las dependencias utilizadas por el DispatcherServlet de Spring en forma de recursos.

  En este fichero configuramos un ViewResolver, que será utilizado por cada Controlador para resolver que recurso debe ser enviado como respuesta, desde un documento html, xml, json, pdf, imagen.

  También se configuran las java annotations para que podamos usar anotaciones en los controladores y así definir que controlador será el encargado de atender que petición para su resolución.

- ***Web.xml***

  Descriptor de despliegue (en inglés Deployment Descriptor) (DD) es un componente de aplicaciones J2EE que describe cómo se debe desplegar (o implantar) una aplicación. Usamos XML para la sintaxis del fichero descriptor de despliegue en aplicaciones J2EE, y para configurar Spring en la aplicación tenemos que declarar DispatcherServlet y ContextLoaderListener en el web.xml.

4.6.2 Ficheros Java Server Pages del modulo de vista

La Carpeta ***/WEB-INF/JSP/*** contiene los siguiente ficheros

- ***selección.jsp:*** Fichero JSP encargado de mostrar el resultado devuelto por el controlador de selección, su función es mostrar el resultado de la transformación y el procesado de un conjunto de datos.
- ***menuLateral.jsp***: Fichero JSP encargado de mostrar el menú de navegación lateral de la aplicación su función es permitir la navegación del usuario entre los diferentes módulos de la aplicación.
- ***preprocesado.jsp***: Fichero JSP encargado de mostrar el resultado devuelto por el controlador de procesado, su función es mostrar el resultado del procesado de un atributo dado un conjunto de datos y un atributo que pertenece a dicho conjunto de datos.
- ***patrones.jsp***: Fichero JSP encargado de mostrar el resultado devuelto por el controlador de patrones, su función es mostrar el resultado del entrenamiento de un algoritmo determinado dado un conjunto de datos y modificando una serie de parámetros para dicho algoritmo.
- ***Evaluacion.jsp***: Fichero JSP encargado de mostrar el resultado devuelto por el controlador de evaluación, su función es mostrar el resultado de la evaluación de un algoritmo determinado dado un conjunto de datos y modificando una serie de parámetros para dicho algoritmo.

La Carpeta ***/WEB-INF/jsp/includes*** contiene los siguiente ficheros:

Ficheros llamados desde ***/jsp/patrones.jsp:***

- ***kmeans.jsp*** Fichero jsp encargado de mostrar el resultado del entrenamiento del algoritmo KMeans.
- ***em.jsp*** Fichero jsp encargado de mostrar el resultado del entrenamiento del algoritmo EM.
- ***naiveBayes.jsp*** Fichero jsp encargado de mostrar el resultado del entrenamiento del algoritmo Naive Bayes.

- ***tree.jsp*** Fichero jsp encargado de mostrar el resultado del entrenamiento del algoritmo C4.5.

Ficheros llamados desde ***/jsp/evaluacion.jsp:***

- ***evaluacionClase.jsp*** Fichero JSP encargado de mostrar el resultado de la evaluación de los algoritmos de clasificación.
- ***evaluacionCluster.jsp*** Fichero JSP encargado de mostrar el resultado de la evaluación de los algoritmos de clustering.

La Carpeta ***/WEB-INF/tag/*** contiene los siguientes ficheros:

- ***nodeTree.tag*** es una plantilla de etiquetas JSTL personalizada y recursiva , su función es mostrar un árbol de nodos en un fichero JSP, es utilizado por el fichero.

  /jsp/ includes/tree.jsp para mostrar el árbol de nodos con su solo pasarle el nodo padre.

4.6.3 Relación de Bibliotecas

En la siguiente tabla tenemos la relación de bibliotecas o JARs externos que se usan en la aplicación y su correspondiente descripción.

Biblioteca	**Descripción**
org.springframework.core	Proporciona clases básicas para el manejo de excepciones y la detección de versiones, y otras clases de ayuda que no están especificados en ninguna parte de la estructura de Spring.
org.springframework.web	Contiene interfaces comunes y genéricas que definen puntos de unión entre la infraestructura Web de Spring y otros Frameworks.
org.springframework.web.servlet	Proporciona Servlets que se integran con la infraestructura de contexto de aplicación, Así como las interfaces y clases principales para el Frameworks MVC Web de Spring.

org.springframework.beans	Esta biblioteca contiene interfaces y clases que permiten a Spring manejar objetos java beans.
org.springframework.context	Este paquete basa su construcción sobre el paquete de Beans para añadir soporte para fuentes de mensajes y para el patrón de diseño tipo Observador, y añade a los objetos de la aplicación la capacidad de obtener recursos utilizando una API consistente.
org.springframework.asm	API que implemente ASM, El ASM es un pequeño Framework que se utiliza para analizar y manipular códigos de bytes de Java. Se utiliza por Spring para modificar dinámicamente el código de bytes de Java y generar el nuevo código byte en tiempo de ejecución.
org.springframework.expression	API que implementa Spring Expression Language (Lenguaje de expresiones de Spring, SpEL) es un lenguaje de expresiones dinámico, SpEL permite la consulta y manipulación de objetos en tiempo real.
commons-io	es una biblioteca de utilidades para ayudar en el desarrollo de la funcionalidad de Input/Output, Además contiene clases de utilidad, implementaciones de flujo, filtros de archivos, comparadores de archivos, clases de transformación, Commons IO es desarrollada por apache software foundation.
commons-logging	Proporciona una API de Logging de Java, Es una biblioteca que implementa la creación, manejo de los logs en java, Commons *Logging* es desarrollada por apache software foundation.
Servlet-API	Proporciona una API que contiene clases e interfaces que implementan funcionalidades de los Servlets y define la relación entre los Servlets y el entorno de ejecución.
JSTL	Java Server Pages Estándar Tag Library (JSTL) Encapsula etiquetas simples como funcionalidad básica común a aplicaciones Web. JSTL tiene soporte para tareas comunes,

	estructurales como la iteración y los condicionales, etiquetas para manipular documentos XML, internacionalización, etiquetas SQL. También proporciona un marco para integrar las etiquetas existentes con las etiquetas JSTL.
Slf4j API	Simple Logging Facade for Java (SLF4J) Proporciona una API de Logging de Java por medio de un patrón de diseño estructural tipo fachada (*Facade pattern*) El backend de Logging se determina en tiempo de ejecución mediante la adición de un complemento de logging y puede ser java.util.logging, log4j o logback.
Hibernate validator	Esta api define un modelo de metadatos para la validación de Beans (Bean Validation), El origen de metadatos predeterminado son anotaciones, con la capacidad de anular y ampliar los metadatos a través del uso de XML. La API no está ligada a un nivel de aplicación específica o modelo de programación. Específicamente no está ligado ni a la capa Web o el nivel de persistencia, y está disponible tanto para la Programación de aplicaciones del lado del servidor, así como los desarrolladores de aplicaciones de cliente.
Validation api	API que implementa la Validación JavaBeans (Bean Validation) es un modelo de validación disponible como parte de la plataforma Java EE 6. El modelo de Bean Validation está soportado por restricciones en forma de anotaciones colocadas en un campo, método o una clase de un componente JavaBeans, como un bean gestionado.
Weka API	(Waikato Environment for Knowledge Analysis - *Entorno para Análisis del Conocimiento de la Universidad de Waikato*), es una API para aprendizaje automático y minería de datos escrita en Java y desarrollada en la Universidad de Waikato. Weka es software libre distribuido bajo licencia GNU-GPL, contiene la implementación de los algoritmos de aprendizaje automático.

Tabla 13 – Relación de bibliotecas externas

4.7. Realización de casos de uso

Antes de empezar a definir los casos de uso de la aplicación, recordamos brevemente lo que se entiende por un caso de uso

Un caso de uso es una descripción de los pasos o las actividades que deberán realizarse para llevar a cabo algún proceso. Los personajes o entidades que participarán en un caso de uso se denominan actores. En el contexto de ingeniería del software, un caso de uso es una secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema.

Los diagramas de casos de uso sirven para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios y/u otros sistemas. O lo que es igual, un diagrama que muestra la relación entre los actores y los casos de uso en un sistema. Una relación es una conexión entre los elementos del modelo, por ejemplo la especialización y la generalización son relaciones. Los diagramas de casos de uso se utilizan para ilustrar los requerimientos del sistema al mostrar cómo reacciona a eventos que se producen en su ámbito o en él mismo.

Los más comunes para la captura de requisitos funcionales, especialmente con el desarrollo del paradigma de la programación orientada a objetos, donde se originaron, si bien puede utilizarse con resultados igualmente satisfactorios con otros paradigmas de programación.

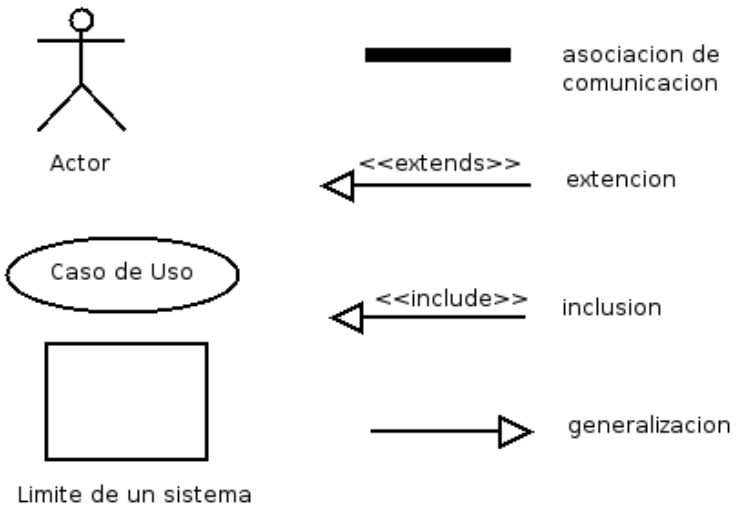


Figura 29 – Notación usada en un caso de uso

Fuente [wikipedia4.7]

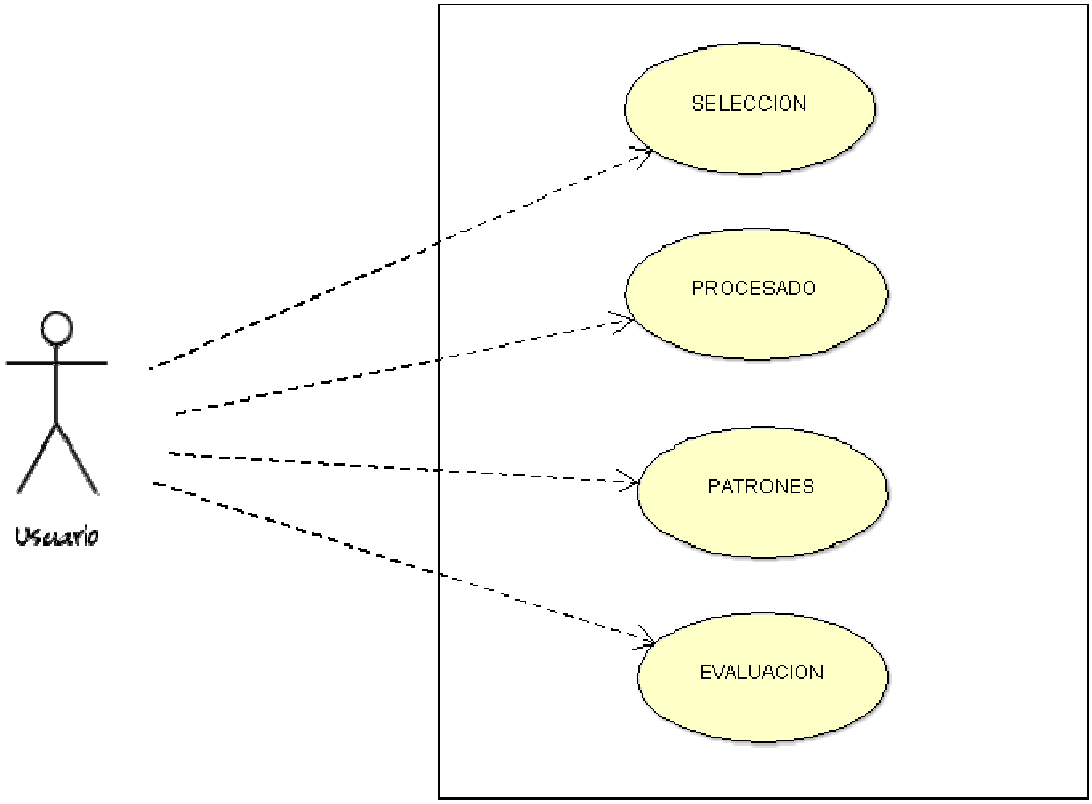


Figura 30 – Diagrama de casos de uso

Caso de uso SELECCIÓN

Selecciona un fichero que contiene un conjunto de datos en formato texto y lo carga en la aplicación para su posterior transformación en objetos.

Caso de uso PROCESADO

Procesa cada uno de los atributos del conjunto de datos dado y nos muestra sus propiedades, tiene que existir un conjunto de datos cargado en la aplicación como requisito para su funcionamiento.

Caso de uso PATRONES

Entrena un algoritmo de clasificación o de Clustering para un conjunto de datos dado, tiene que existir un conjunto de datos cargado en la aplicación como requisito para su funcionamiento.

Caso de uso EVALUACION

Evalúa algoritmos de clasificación y Clustering para un conjunto de datos dado, tiene que existir un conjunto de datos cargado en la aplicación como requisito para su funcionamiento.

4.8. Diagrama de paquetes

En el siguiente diagrama vemos la distribución de los paquetes de clases de la aplicación, también vemos la relación entre dichos paquetes, es decir las importaciones y dependencias que tiene cada paquete con los demás.

El único paquete que puede conectar con la API de implementación de algoritmos es el connectAPI, Mientras el paquete BO encargado de la lógica de negocio no necesita importar ni usar la API de implementación de algoritmos, ya que implementa las operaciones que necesita a través de métodos y clases del connectAPI de manera transparente sin depender de la implementación interna de las clases del connectAPI, y sin saber qué tipo de API se está usando.

Con lo que connectAPI hace de capa de transparencia o separación para que nuestra aplicación no dependa de la API de implementación de algoritmos.

Los demás paquetes tienen dependencias entre si y no pueden acceder directamente a la API de implementación de algoritmos sin pasar por el connectAPI.

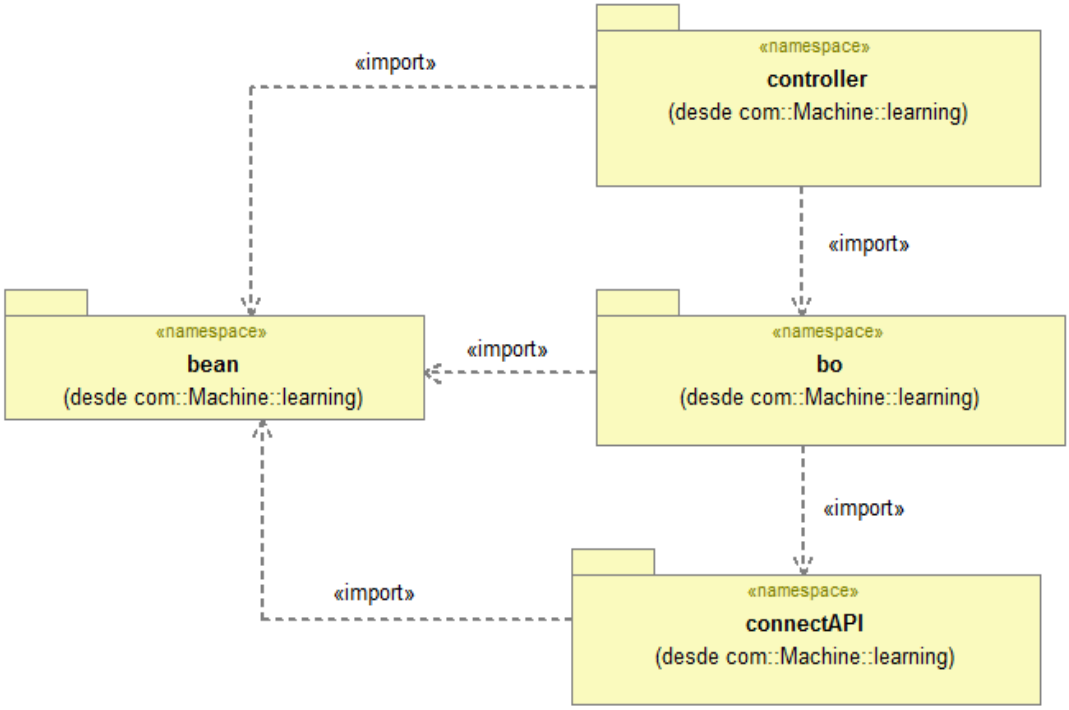


Figura 31 – Diagrama de paquetes

4.9. Diagrama de Clases

Es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargaran del funcionamiento y la relación entre uno y otro, a continuación se muestra a modo de ejemplo una de las clases para poder ver en detalle cómo se definen los atributos y operaciones.

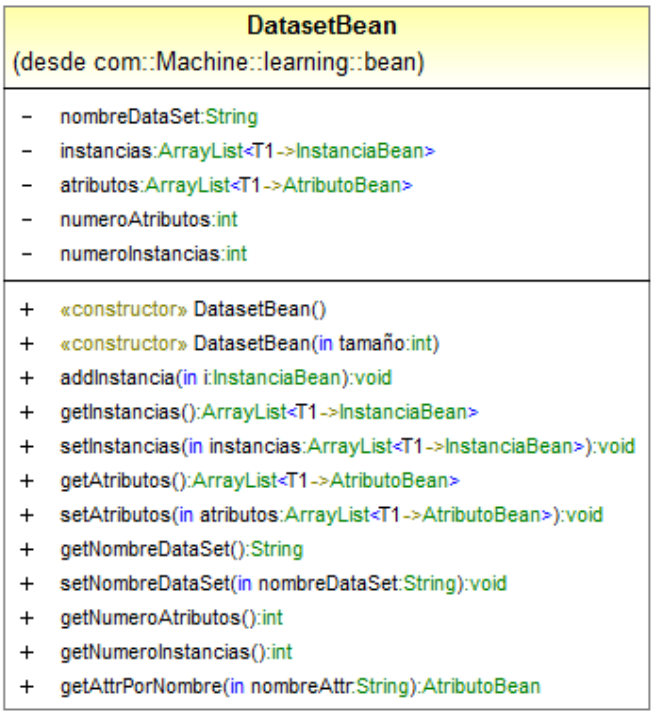


Figura 32 – Diagrama de una clase UML

A continuación se muestran los diagramas de clases por cada uno de los casos de usos definidos anteriormente en el apartado 5.7, se mostraran cuatro diagrama de clases que corresponden a los cuatro casos de uso: selección, procesado, patrones y evaluación,

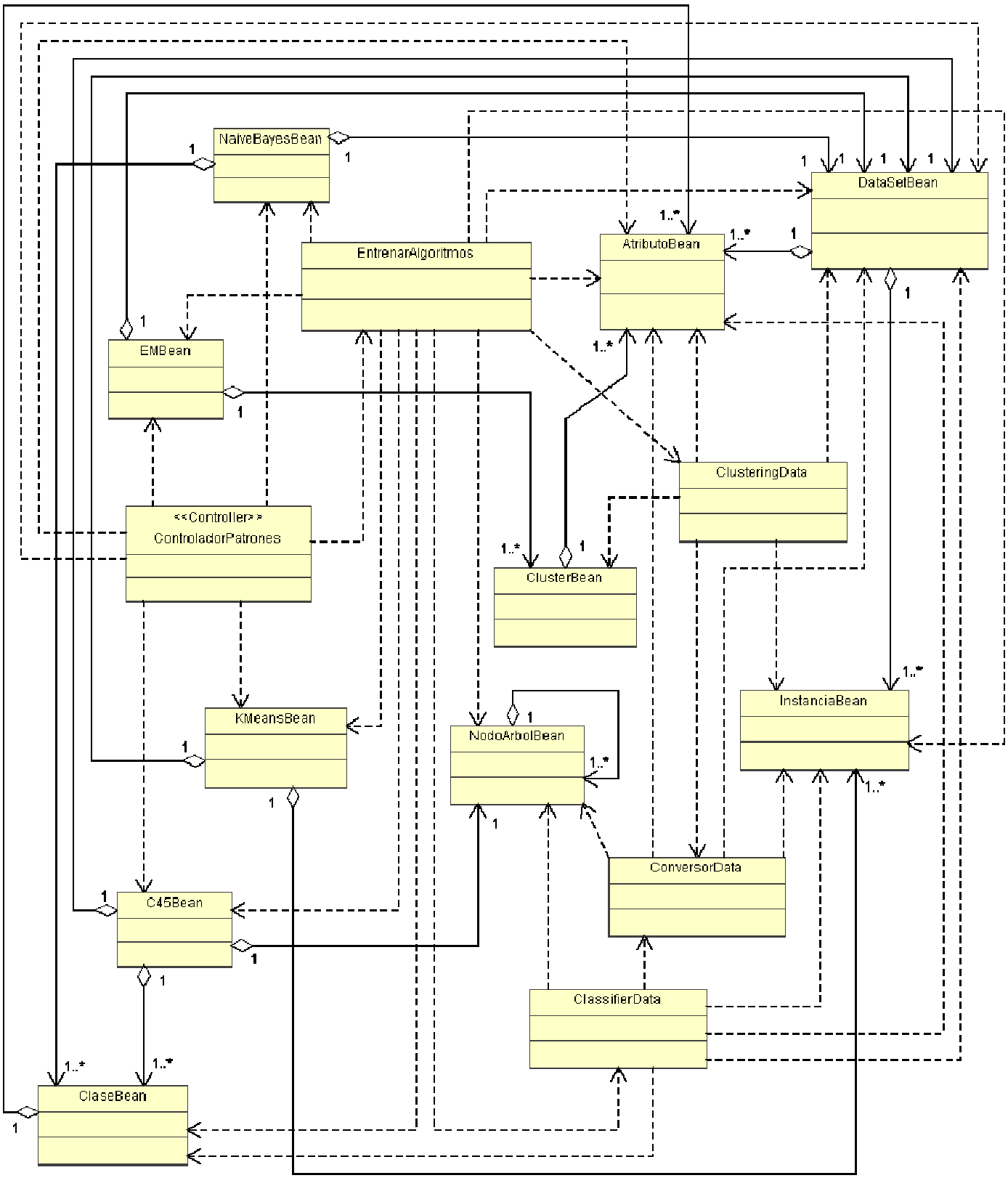
Para los casos de uso de selección y procesado se mostraran todos los atributos y métodos y conexiones que hay entre las distintas clases,

Para los casos de uso de patrones y evaluación y debido a la gran cantidad de atributos y métodos de las clases implicadas, hemos optado por un modelo más simplificado en el que no aparecerán los atributos ni las operaciones de las diferentes clases para que se pueda mostrar el diagrama completo.

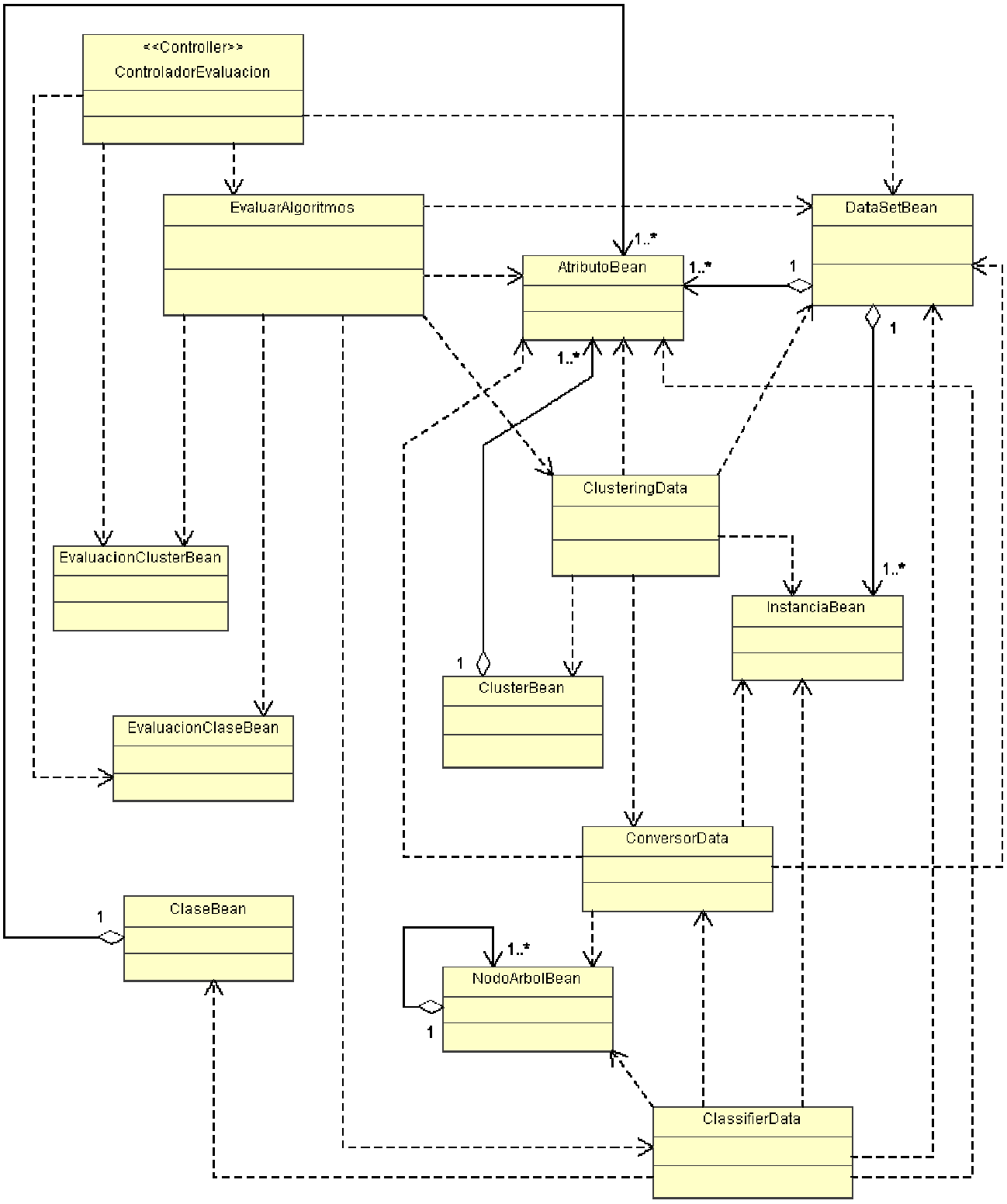
4.9.1 DIAGRAMA DE CLASES CASO DE USO SELECCIÓN - Figura 33

4.9.2 DIAGRAMA DE CLASES CASO DE USO PROCESADO - Figura 34

4.9.3 DIAGRAMA DE CLASES CASO DE USO PATRONES - Figura 35



4.9.4 DIAGRAMA DE CLASES CASO DE USO EVALUACION - Figura 36



4.10 Guía para incorporar un nuevo algoritmo a la aplicación

El objetivo de este apartado es servir como una guía para la inclusión de un nuevo algoritmo en la plataforma desarrollada. La guía establece los pasos necesarios para la incorporación de un nuevo algoritmo y realizar los procesos de entrenamiento y evaluación con diversos conjuntos de datos de la misma forma que se hace con los demás algoritmos ya existentes en la aplicación. Los pasos necesarios para ello se listan a continuación:

1. Crear un nuevo objeto JAVA BEAN

***Paquete**: com.machine.learning.bean*

Crear un nuevo objeto java bean que almacenará los resultados del entrenamiento y evaluación del algoritmo nuevo. El bean creado tendrá como mínimo los siguientes atributos:

- ***DatasetBean***: java bean para almacenar el conjunto de datos utilizado en el entrenamiento y evaluación del algoritmo.
- Numero de clústeres o de clases dependiendo si el algoritmo nuevo es de clasificación o de agrupamiento.
- Listado de objetos ***AtributoBean***: listado de los atributos del conjunto de datos utilizado en el entrenamiento o evaluación del algoritmo nuevo.
- Listado de clases o clústeres dependiendo de si el algoritmo añadido es de clasificación o agrupamiento.
- Nombre del algoritmo.

A parte de las variables descritas anteriormente, se añadirán variables específicas del algoritmo nuevo. Este objeto le asignamos un nombre tipo XXBean donde XX es el código que le asignamos al algoritmo nuevo, y que posteriormente utilizaremos en los pasos siguientes.

2. Añadir un nuevo método para entrenar el algoritmo

***Clase**: com.machine.learning.bo.EntrenarAlgoritmos*

Este nuevo método se encargara de entrenar el algoritmo nuevo, utilizando un conjunto de datos dado, para luego devolver un java bean que almacene el resultado del entrenamiento. El método creado tiene que cumplir el siguiente formato:

Parámetros de entrada:

- ***DatasetBean***: java bean que almacena el conjunto de datos que utilizaremos en el entrenamiento del algoritmo nuevo.
- listado de objetos ***AtributoBean***: un listado que contiene todos los atributos del conjunto de datos que utilizaremos en el entrenamiento del algoritmo nuevo.
- Clase índice: este parámetro indica que atributo será utilizado como clase discriminatoria o índice, solo para algoritmos de clasificación.
- Número máximo de iteraciones: indica el número máximo de iteraciones permitido.
- Numero clústeres: variable para indicar el numero de clústeres al iniciar el entrenamiento , solo para algoritmo de clustering o agrupamiento

El nuevo método creado tiene que devolver un objeto de la clase **XXBean,** como se ha mencionado anteriormente en el paso 1. XX es el código que le asignamos al algoritmo nuevo.

3. Añadir un nuevo método para evaluar el algoritmo.

Clase : com.machine.learning.bo.EvaluarAlgoritmos

Este nuevo método se encargara de evaluar el algoritmo nuevo, utilizando un conjunto de datos dado, para luego devolver un java bean que almacene el resultado de la evaluación. El método creado tiene que cumplir el siguiente formato:

Parámetros de entrada:

- ***DatasetBean***: java bean que almacena el conjunto de datos que utilizaremos en la evaluación del algoritmo nuevo.
- Clase índice: este atributo indica que atributo será utilizado como clase discriminatoria o índice, solo para algoritmos de clasificación.
- Nombre del algoritmo a evaluar.

El nuevo método creado tiene que devolver un objeto de la clase ***EvaluacionClusterBean*** en caso de que el algoritmo nuevo sea de agrupamiento (clustering) o un objeto de la clase ***EvaluacionClaseBean*** en caso de que sea un algoritmo de clasificación.

4. Crear y modificar vistas de presentación JSP

Crear una vista de presentación JSP. La vista creada se utilizara para visualizar los resultados del entrenamiento del algoritmo nuevo. Este fichero lo llamaremos XX.jsp, siendo XX es el código que le hemos asignado al algoritmo nuevo en el paso 1.

Ubicación de la vista de presentación ***/jsp/includes/XX.jsp***

Para crear dicha vista se puede tomar como referencia una de las vistas ya existentes, con eso podemos tener los elementos de presentación comunes como listados de conjunto de datos, atributos, etc. Para lo demás adaptamos la vista según las necesidades de lo que queremos visualizar cuando entrenamos el algoritmo nuevo. Posteriormente tenemos que llamar la vista creada desde la página ***/jsp/patrones.jsp.*** Para ello añadimos una referencia para la invocación de la vista de presentación creada anteriormente.

Para la presentación de resultados de evaluación del algoritmo nuevo no será necesario crear una vista nueva o modificar alguna de las dos vistas ya existentes:

- Para algoritmos de clasificación: ***/jsp/evaluacionClase.jsp***
- Para algoritmos de clustering: ***/jsp/evaluacionCluster.jsp***

Ahora que ya tenemos las dos vistas completas, el siguiente paso es adaptar los selectores que hay tanto en la vistas de entrenamiento como los de las vistas de evaluación. Para ello añadimos una entrada nueva en los selectores, hay un selector por vista encargado de la llamada de los algoritmos, las vistas encargadas de ello son las siguientes:

- ***/jsp /patrones.jsp***
- ***/jsp/evaluacion.jsp***

Para la nueva entrada en los selectores se usara el código XX que hemos asignado al algoritmo nuevo en el primer paso.

5. Modificar los controladores para llamar a los métodos de entrenamiento y evaluación.

Clase: com.machine.learning.controller.ControladorPatrones

En esta clase tenemos que hacer las modificaciones necesarias para conectar la vista de presentación JSP que hemos creado en el punto 4, con el nuevo método creado en la clase ***com.machine.learning.bo.EntrenarAlgoritmos*** en el punto 2.

Es decir lo que estamos haciendo es conectar los selectores de las páginas de presentación con los métodos correspondientes de entrenamiento. Para ello tenemos que añadir una instrucción de control (IF) usando el mismo código XX que hemos utilizado en el paso 1.

***Clase**: com.machine.learning.controller.ControladorEvaluacion*

En esta clase tenemos que hacer las modificaciones necesarias para conectar la vista de presentación JSP que hemos creado en el punto 4, con el nuevo método creado en la clase ***com.machine.learning.bo.EvaluarAlgoritmos*** en el punto 3.

Es decir lo que estamos haciendo es conectar los selectores de las páginas de presentación con los métodos correspondientes de evaluación. Para ello únicamente tenemos que añadir una instrucción de control (IF) usando el mismo código XX que hemos utilizado en el paso 1.

Capítulo 5

5. Pruebas

El objetivo de este capítulo es comprobar si la herramienta que se ha desarrollado cumple con un rendimiento aceptable desde el primer paso , tanto en la adquisición de la información como su transformación y procesado , además de comprobar el correcto funcionamiento de la implementación de los algoritmos y diversos métodos para analizar la información y la obtención de resultados y patrones , con este objetivo en mente, se han definido una serie de pruebas para validar el correcto funcionamiento de la herramienta.

Para hacer las correspondientes pruebas usaremos dos conjuntos de datos y sus atributos que vamos describiendo a continuación:

1. Conjunto de datos weather

Conjunto de datos que contiene 14 instancias o muestras y 5 atributos (jugar, pronostico, temperatura, humedad, viento), que describe la posibilidad de jugar o no jugar al tenis según las condiciones meteorológicas de cada caso, Fuente [weka5].

```
@relation weather


@attribute outlook {sunny, overcast, rainy}
@attribute temperature real
@attribute humidity real
@attribute windy {TRUE, FALSE}
@attribute play {yes, no}


@data
sunny,85,85,FALSE,no
sunny,80,90,TRUE,no
overcast,83,86,FALSE,yes
rainy,70,96,FALSE,yes
rainy,68,80,FALSE,yes
rainy,65,70,TRUE,no
overcast,64,65,TRUE,yes
sunny,72,95,FALSE,no
sunny,69,70,FALSE,yes
rainy,75,80,FALSE,yes
sunny,75,70,TRUE,yes
overcast,72,90,TRUE,yes
overcast,81,75,FALSE,yes
rainy,71,91,TRUE,no
```

2. Conjunto de datos iris

Conjunto de datos que contiene 30 instancias o muestras y 5 atributos (longitud y anchura del pétalo, longitud y anchura del sépalo) que describen de que especie de iris es una planta según sus características, Fuente [weka5].

```
@RELATION iris

@ATTRIBUTE sepallength  REAL
@ATTRIBUTE sepalwidth   REAL
@ATTRIBUTE petallength  REAL
@ATTRIBUTE petalwidth   REAL
@ATTRIBUTE grupo {Iris-setosa,Iris-versicolor,Iris-virginica}

@DATA
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
6.9,3.1,4.9,1.5,Iris-versicolor
5.5,2.3,4.0,1.3,Iris-versicolor
6.5,2.8,4.6,1.5,Iris-versicolor
6.3,3.3,4.7,1.6,Iris-versicolor
4.9,2.4,3.3,1.0,Iris-versicolor
6.6,2.9,4.6,1.3,Iris-versicolor
5.9,3.0,4.2,1.5,Iris-versicolor
6.0,2.2,4.0,1.0,Iris-versicolor
6.1,2.9,4.7,1.4,Iris-versicolor
5.6,2.9,3.6,1.3,Iris-versicolor
7.1,3.0,5.9,2.1,Iris-virginica
6.3,2.9,5.6,1.8,Iris-virginica
6.5,3.0,5.8,2.2,Iris-virginica
7.6,3.0,6.6,2.1,Iris-virginica
4.9,2.5,4.5,1.7,Iris-virginica
6.5,3.2,5.1,2.0,Iris-virginica
6.4,2.7,5.3,1.9,Iris-virginica
6.8,3.0,5.5,2.1,Iris-virginica
5.7,2.5,5.0,2.0,Iris-virginica
5.8,2.8,5.1,2.4,Iris-virginica
```

5.1 Conjunto de pruebas de transformación y procesado de datos y atributos

En este apartado veremos pruebas correspondientes tanto a la transformación y procesado de la información, como su posterior presentación y visualización por parte del usuario, en total veremos dos pruebas descritas a continuación:

1. Prueba 1 - Transformación y Procesado de datos
2. Prueba 2 - Analizar Propiedades de atributos

Nombre	**Prueba 1 - Transformación y Procesado de datos**
Descripción	Procesado de los atributos y datos contenidos en un fichero de texto y su posterior transformación en objetos java legibles por la aplicación.
Entrada	Fichero de texto que contiene los atributos y un conjunto de datos (conjunto de datos weather)
Salida	Un conjunto de objetos java legibles por la aplicación y posterior presentación en formato HTML y CSS legibles por el usuario.
Proceso de Prueba	Seleccionar un fichero de texto con formato ARFF compatible y cargarlo en la aplicación, la aplicación se encarga de transformarlo procesarlo y presentarlo.
Resultado obtenido	Atributos y conjunto de datos presentados en tablas
Resultado Esperado	Atributos y conjunto de datos presentados en tablas
Veredicto	Pasa la prueba

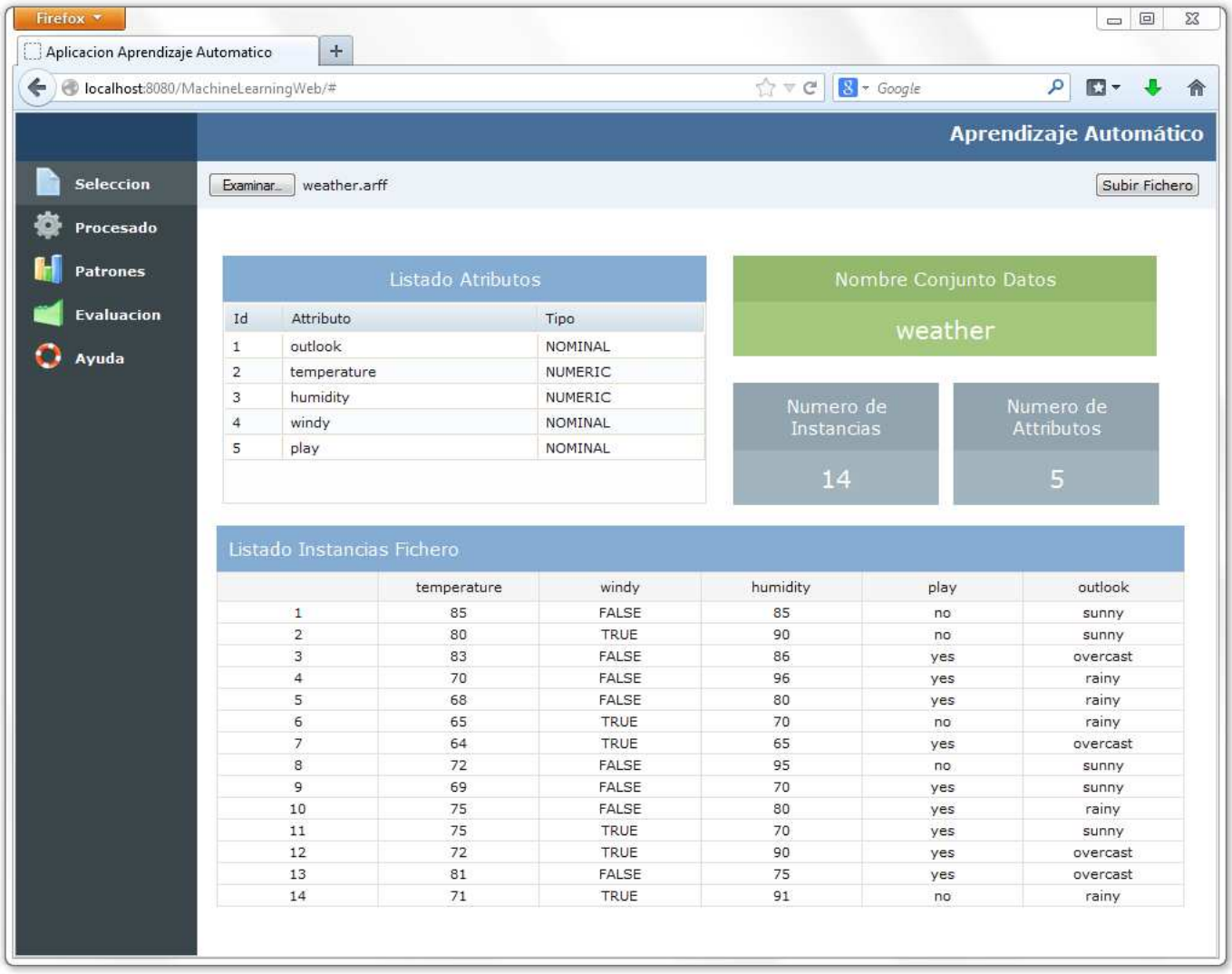


Figura 37 – Prueba selección datos

Nombre	**Prueba 2 - Analizar propiedades de atributos**
Descripción	Analiza las propiedades de un determinado atributo
Entrada	Conjunto de datos y atributos, seleccionamos un determinado atributo a analizar a través de un campo selector.
Salida	Propiedades del atributo seleccionado y su presentación en formato HTML y CSS legibles por el usuario.
Requisitos	Tener un conjunto de datos y atributos cargado en la aplicación (conjunto de datos weather)
Proceso de Prueba	Seleccionar un determinado atributo , la aplicación procesa dicha

	información y devuelve las propiedades de dicho atributos
Resultado obtenido	Propiedades de un atributo: tipo, media, máximo, mínimo, desviación estándar
Resultado Esperado	Propiedades de un atributo: tipo, media, máximo, mínimo, desviación estándar
Veredicto	Pasa la prueba

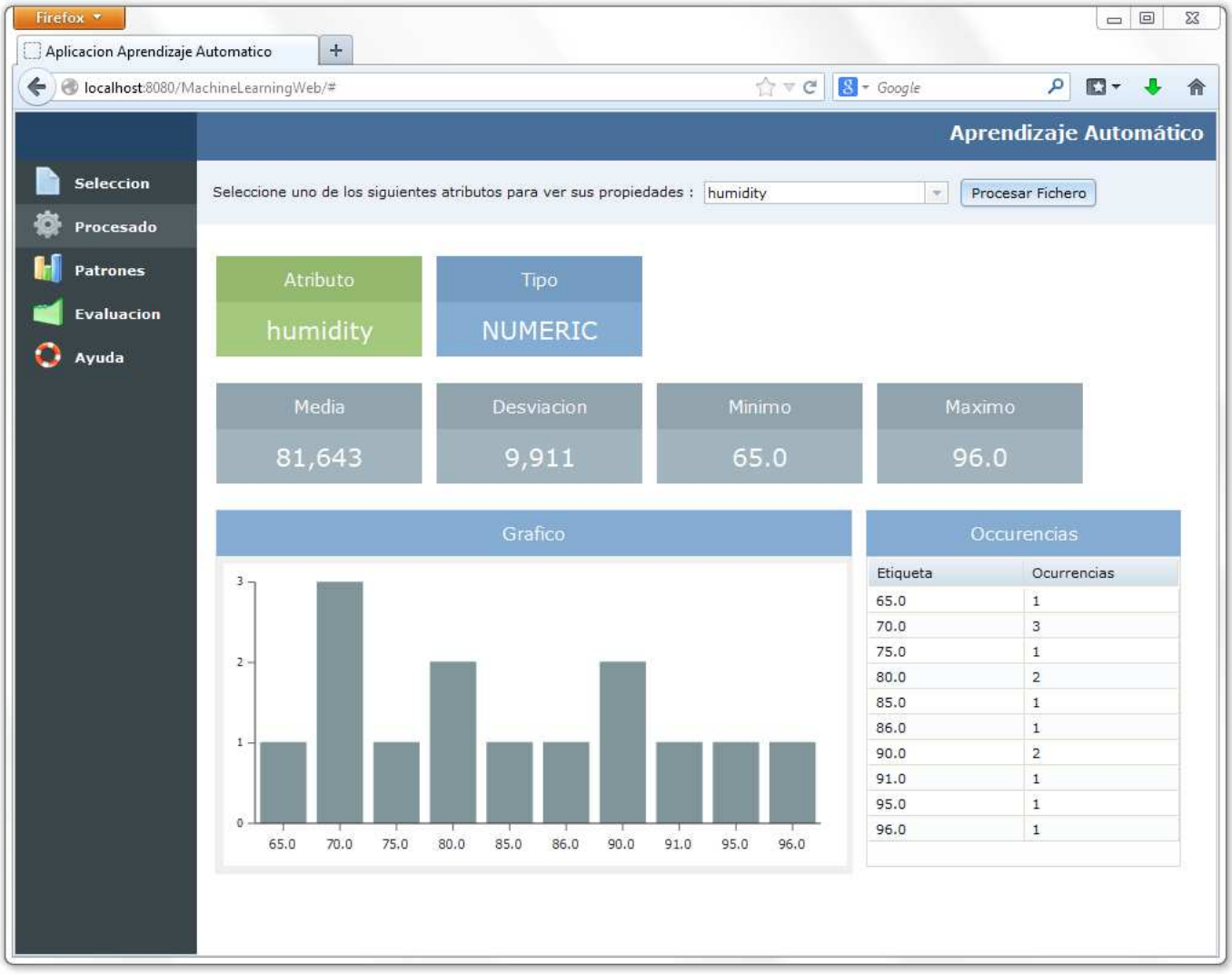


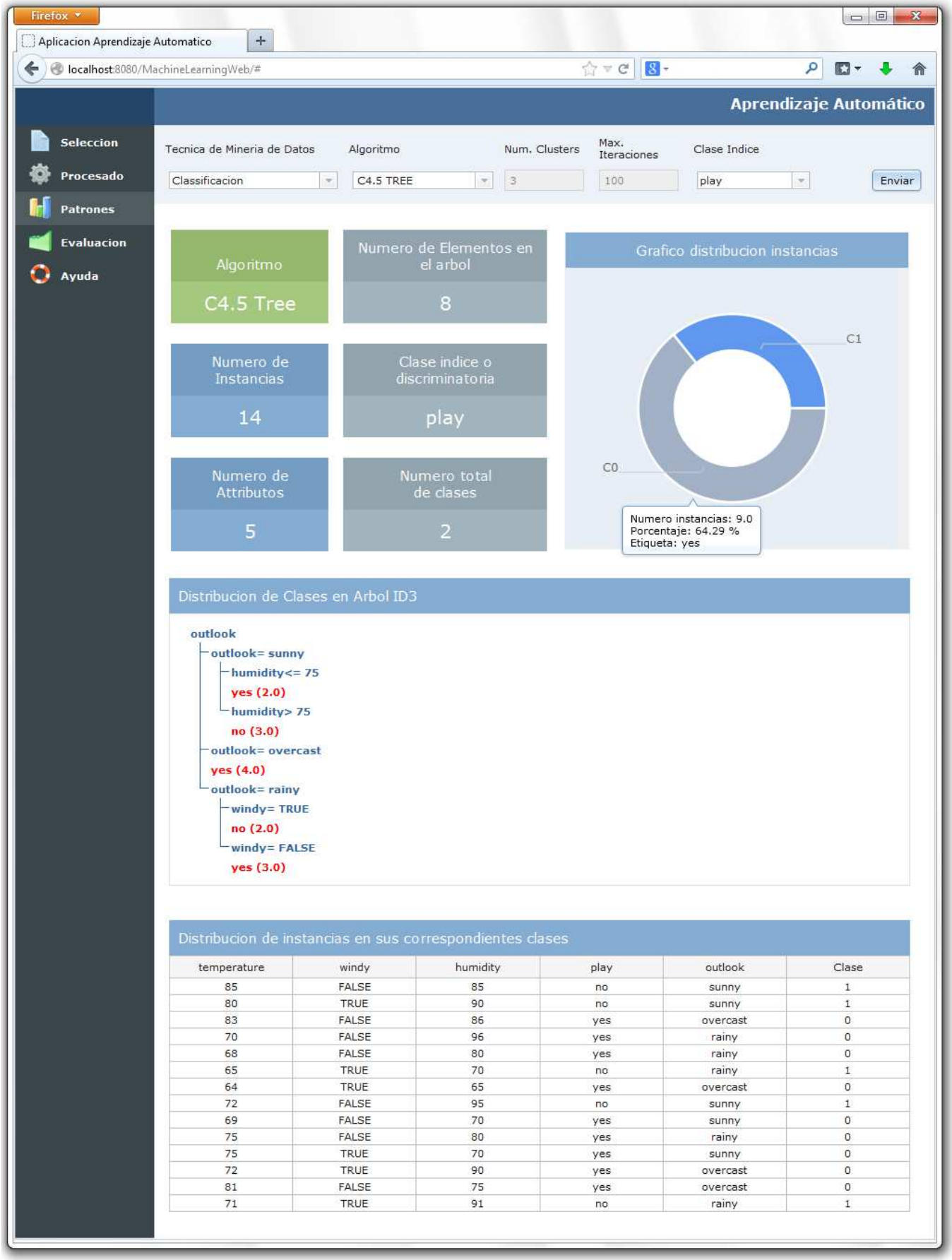
Figura 38 - Prueba propiedades de atributos

5.2 Conjunto de pruebas de entrenamiento de algoritmos

En este apartado veremos las pruebas correspondientes al entrenamiento de los distintos algoritmos de aprendizaje automático que soporta la aplicación, como su posterior presentación y visualización por parte del usuario, En total veremos cuatro pruebas descritas a continuación

1. Prueba 3 - Entrenamiento del algoritmo de árbol C4.5
2. Prueba 4 - Entrenamiento del algoritmo Naive Bayes
3. Prueba 5 - Entrenamiento del algoritmo KMENAS
4. Prueba 6 - Entrenamiento del algoritmo EM

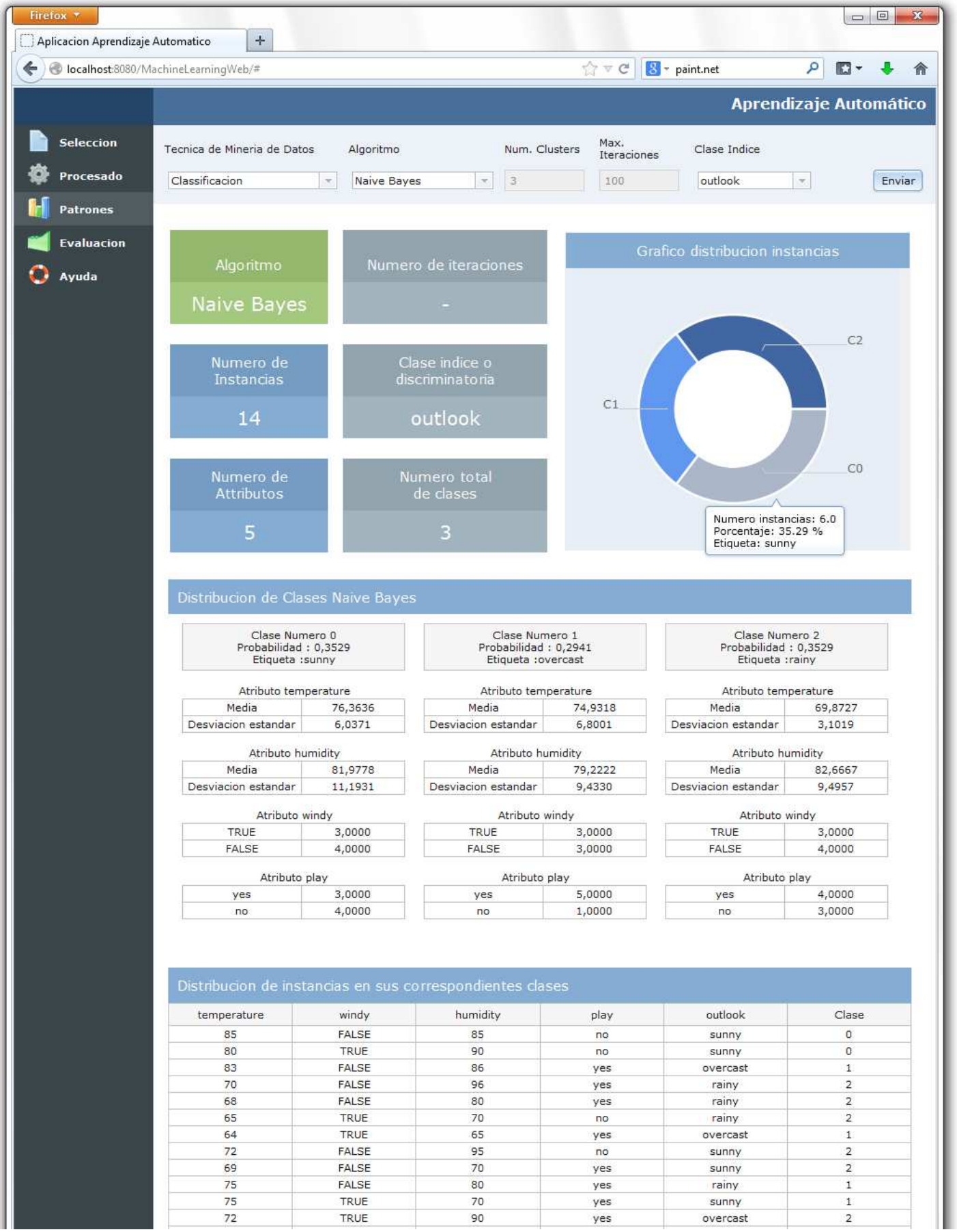
Nombre	**Prueba 3 - Entrenamiento del algoritmo de árbol C4.5**
Descripción	Entrenamiento del algoritmo C4.5
Entrada	Conjunto de datos y atributos debidamente procesado y transformado
Salida	Representación textual y grafica de los resultados obtenidos del entrenamiento del algoritmo C4.5
Requisitos	Tener un conjunto de datos y atributos cargado en la aplicación (conjunto de datos weather)
Proceso de Prueba	Cargamos un fichero de texto que contenga un conjunto de datos y atributos valido y lo procesamos con la aplicación para luego entrenar dichos datos con el algoritmo C4.5, Existe la posibilidad de cambiar la clase índice a través de un campo selector. La aplicación procesa dicha información y devuelve una representación textual y grafica del resultado del entrenamiento de los datos con dicho algoritmo.
Resultado esperado	Número total de clases, número de elementos del árbol Representación grafica de un árbol de discriminación según la clase índice de entrada, distribución correcta de las instancias en sus correspondientes clases y su representación gráfica.
Resultado obtenido	El resultado obtenido coincide con el resultado esperado.
Veredicto	Pasa la prueba



temperature	windy	humidity	play	outlook	Clase
85	FALSE	85	no	sunny	1
80	TRUE	90	no	sunny	1
83	FALSE	86	yes	overcast	0
70	FALSE	96	yes	rainy	0
68	FALSE	80	yes	rainy	0
65	TRUE	70	no	rainy	1
64	TRUE	65	yes	overcast	0
72	FALSE	95	no	sunny	1
69	FALSE	70	yes	sunny	0
75	FALSE	80	yes	rainy	0
75	TRUE	70	yes	sunny	0
72	TRUE	90	yes	overcast	0
81	FALSE	75	yes	overcast	0
71	TRUE	91	no	rainy	1

Figura 39 – Prueba entrenamiento algoritmo C4.5

Nombre	Prueba 4 - Entrenamiento del algoritmo Naive Bayes
Descripción	Entrenamiento del algoritmo Naive Bayes
Entrada	Conjunto de datos y atributos debidamente procesado y transformado
Salida	Representación textual y grafica de los resultados obtenidos del entrenamiento del algoritmo Naive Bayes
Requisitos	Tener un conjunto de datos y atributos cargado en la aplicación (conjunto de datos weather)
Proceso de Prueba	Cargamos un fichero de texto que contenga un conjunto de datos y atributos valido y lo procesamos con la aplicación para luego entrenar dichos datos con el algoritmo Naive Bayes, Existe la posibilidad de cambiar la clase índice a través de un campo selector, La aplicación procesa dicha información y devuelve una representación textual y grafica del resultado del entrenamiento de los datos con dicho algoritmo.
Resultado esperado	Número total de clases Distribución de las clases según el algoritmo Navie Bayes, para cada clase de la distribución aparece su probabilidad y para cada atributo de una clase aparece su media y desviación estándar en caso de que sea un atributo numérico o número de ocurrencias de las etiquetas en caso de que sea un atributo nominal. Distribución de las instancias en sus correspondientes clases y su representación gráfica.
Resultado obtenido	El resultado obtenido coincide con el resultado esperado.
Veredicto	Pasa la prueba



Distribucion de Clases Naive Bayes

Clase Numero 0 Probabilidad : 0,3529 Etiqueta :sunny	Clase Numero 1 Probabilidad : 0,2941 Etiqueta :overcast	Clase Numero 2 Probabilidad : 0,3529 Etiqueta :rainy

Atributo temperature		Atributo temperature		Atributo temperature	
Media	76,3636	Media	74,9318	Media	69,8727
Desviacion estandar	6,0371	Desviacion estandar	6,8001	Desviacion estandar	3,1019

Atributo humidity		Atributo humidity		Atributo humidity	
Media	81,9778	Media	79,2222	Media	82,6667
Desviacion estandar	11,1931	Desviacion estandar	9,4330	Desviacion estandar	9,4957

Atributo windy		Atributo windy		Atributo windy	
TRUE	3,0000	TRUE	3,0000	TRUE	3,0000
FALSE	4,0000	FALSE	3,0000	FALSE	4,0000

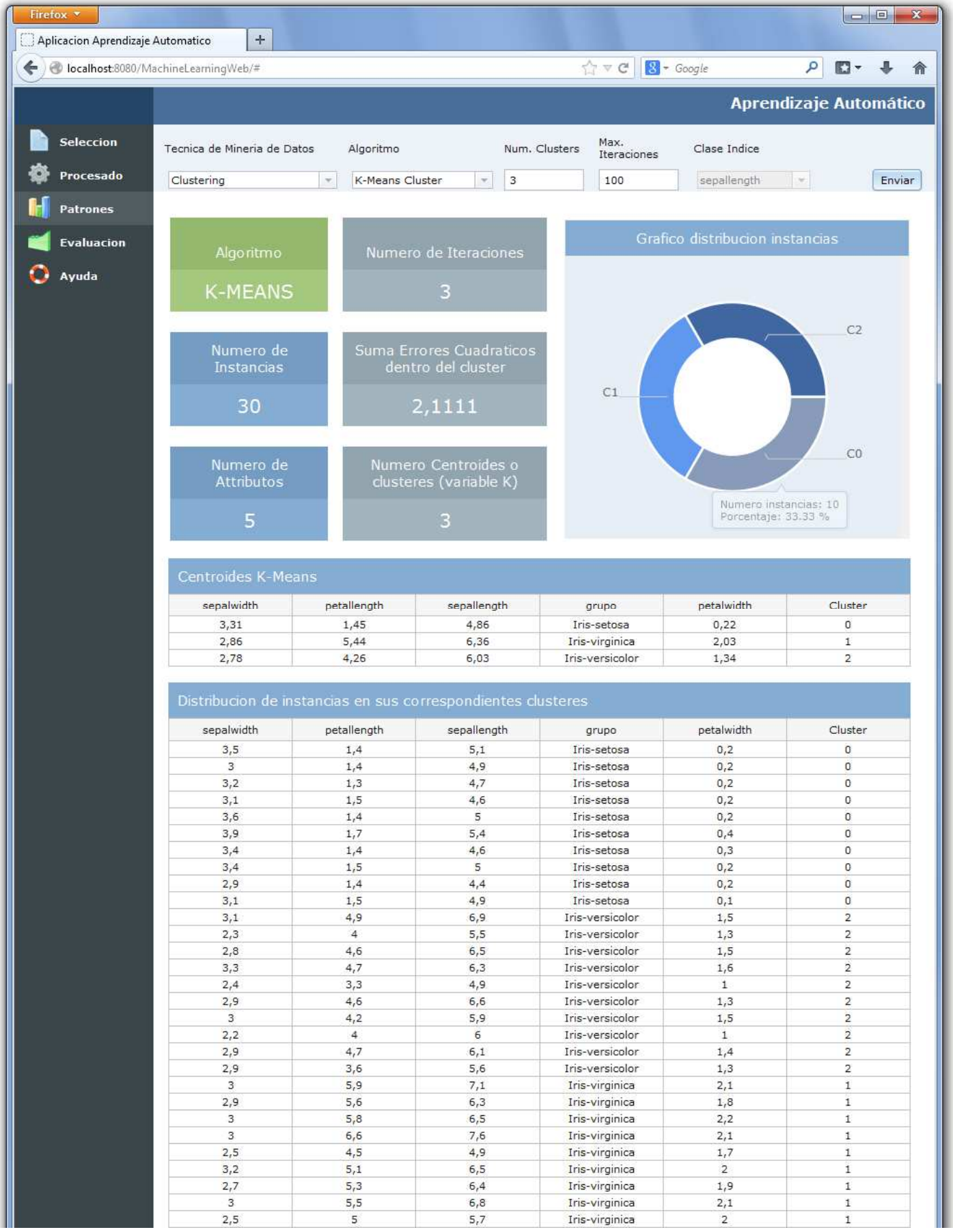
Atributo play		Atributo play		Atributo play	
yes	3,0000	yes	5,0000	yes	4,0000
no	4,0000	no	1,0000	no	3,0000

Distribucion de instancias en sus correspondientes clases

temperature	windy	humidity	play	outlook	Clase
85	FALSE	85	no	sunny	0
80	TRUE	90	no	sunny	0
83	FALSE	86	yes	overcast	1
70	FALSE	96	yes	rainy	2
68	FALSE	80	yes	rainy	2
65	TRUE	70	no	rainy	2
64	TRUE	65	yes	overcast	1
72	FALSE	95	no	sunny	2
69	FALSE	70	yes	sunny	2
75	FALSE	80	yes	rainy	1
75	TRUE	70	yes	sunny	1
72	TRUE	90	yes	overcast	2

Figura 40 – Prueba entrenamiento algoritmo Naive Bayes

Nombre	Prueba 5 - Entrenamiento del algoritmo K-Means
Descripción	Entrenamiento del algoritmo K-Means
Entrada	Conjunto de datos y atributos debidamente procesado y transformado
Salida	Representación textual y grafica de los resultados obtenidos del entrenamiento del algoritmo K-Means
Requisitos	Tener un conjunto de datos y atributos cargado en la aplicación (conjunto de datos IRIS)
Proceso de Prueba	Cargamos un fichero de texto que contenga un conjunto de datos y atributos valido y lo procesamos con la aplicación para luego entrenar dichos datos con el algoritmo K-Means, El número de clústeres y el número máximo de iteraciones son dos variables de entrada que podemos modificar según el caso, La aplicación procesa dicha información y devuelve una representación textual y grafica del resultado del entrenamiento de los datos con dicho algoritmo.
Resultado esperado	Número iteraciones necesarios para llegar al resultado Suma de errores cuadráticos dentro del clúster Distribución de cada uno de los centroides K-Means. Distribución de las instancias en sus correspondientes clústeres y su representación gráfica.
Resultado obtenido	El resultado obtenido coincide con el resultado esperado.
Veredicto	Pasa la prueba



Centroides K-Means

sepalwidth	petallength	sepallength	grupo	petalwidth	Cluster
3,31	1,45	4,86	Iris-setosa	0,22	0
2,86	5,44	6,36	Iris-virginica	2,03	1
2,78	4,26	6,03	Iris-versicolor	1,34	2

Distribucion de instancias en sus correspondientes clusteres

sepalwidth	petallength	sepallength	grupo	petalwidth	Cluster
3,5	1,4	5,1	Iris-setosa	0,2	0
3	1,4	4,9	Iris-setosa	0,2	0
3,2	1,3	4,7	Iris-setosa	0,2	0
3,1	1,5	4,6	Iris-setosa	0,2	0
3,6	1,4	5	Iris-setosa	0,2	0
3,9	1,7	5,4	Iris-setosa	0,4	0
3,4	1,4	4,6	Iris-setosa	0,3	0
3,4	1,5	5	Iris-setosa	0,2	0
2,9	1,4	4,4	Iris-setosa	0,2	0
3,1	1,5	4,9	Iris-setosa	0,1	0
3,1	4,9	6,9	Iris-versicolor	1,5	2
2,3	4	5,5	Iris-versicolor	1,3	2
2,8	4,6	6,5	Iris-versicolor	1,5	2
3,3	4,7	6,3	Iris-versicolor	1,6	2
2,4	3,3	4,9	Iris-versicolor	1	2
2,9	4,6	6,6	Iris-versicolor	1,3	2
3	4,2	5,9	Iris-versicolor	1,5	2
2,2	4	6	Iris-versicolor	1	2
2,9	4,7	6,1	Iris-versicolor	1,4	2
2,9	3,6	5,6	Iris-versicolor	1,3	2
3	5,9	7,1	Iris-virginica	2,1	1
2,9	5,6	6,3	Iris-virginica	1,8	1
3	5,8	6,5	Iris-virginica	2,2	1
3	6,6	7,6	Iris-virginica	2,1	1
2,5	4,5	4,9	Iris-virginica	1,7	1
3,2	5,1	6,5	Iris-virginica	2	1
2,7	5,3	6,4	Iris-virginica	1,9	1
3	5,5	6,8	Iris-virginica	2,1	1
2,5	5	5,7	Iris-virginica	2	1

Figura 41 – Prueba entrenamiento algoritmo K-Means

Nombre	Prueba 6 - Entrenamiento del algoritmo EM
Descripción	Entrenamiento del algoritmo EM
Entrada	Conjunto de datos y atributos debidamente procesado y transformado
Salida	Representación textual y grafica de los resultados obtenidos del entrenamiento del algoritmo EM
Requisitos	Tener un conjunto de datos y atributos cargado en la aplicación (conjunto de datos IRIS)
Proceso de Prueba	Cargamos un fichero de texto que contenga un conjunto de datos y atributos valido y lo procesamos con la aplicación para luego entrenar dichos datos con el algoritmo EM, El número de clústeres y el número máximo de iteraciones son dos variables de entrada que podemos modificar según el caso, La aplicación procesa dicha información y devuelve una representación textual y grafica del resultado del entrenamiento de los datos con dicho algoritmo.
Resultado esperado	Número iteraciones necesarios para llegar al resultado Cociente de verosimilitud LogLikelihood Distribución de cada uno de los clústeres y sus atributos. Distribución de las instancias en sus correspondientes clústeres y su representación gráfica.
Resultado obtenido	El resultado obtenido coincide con el resultado esperado.
Veredicto	Pasa la prueba

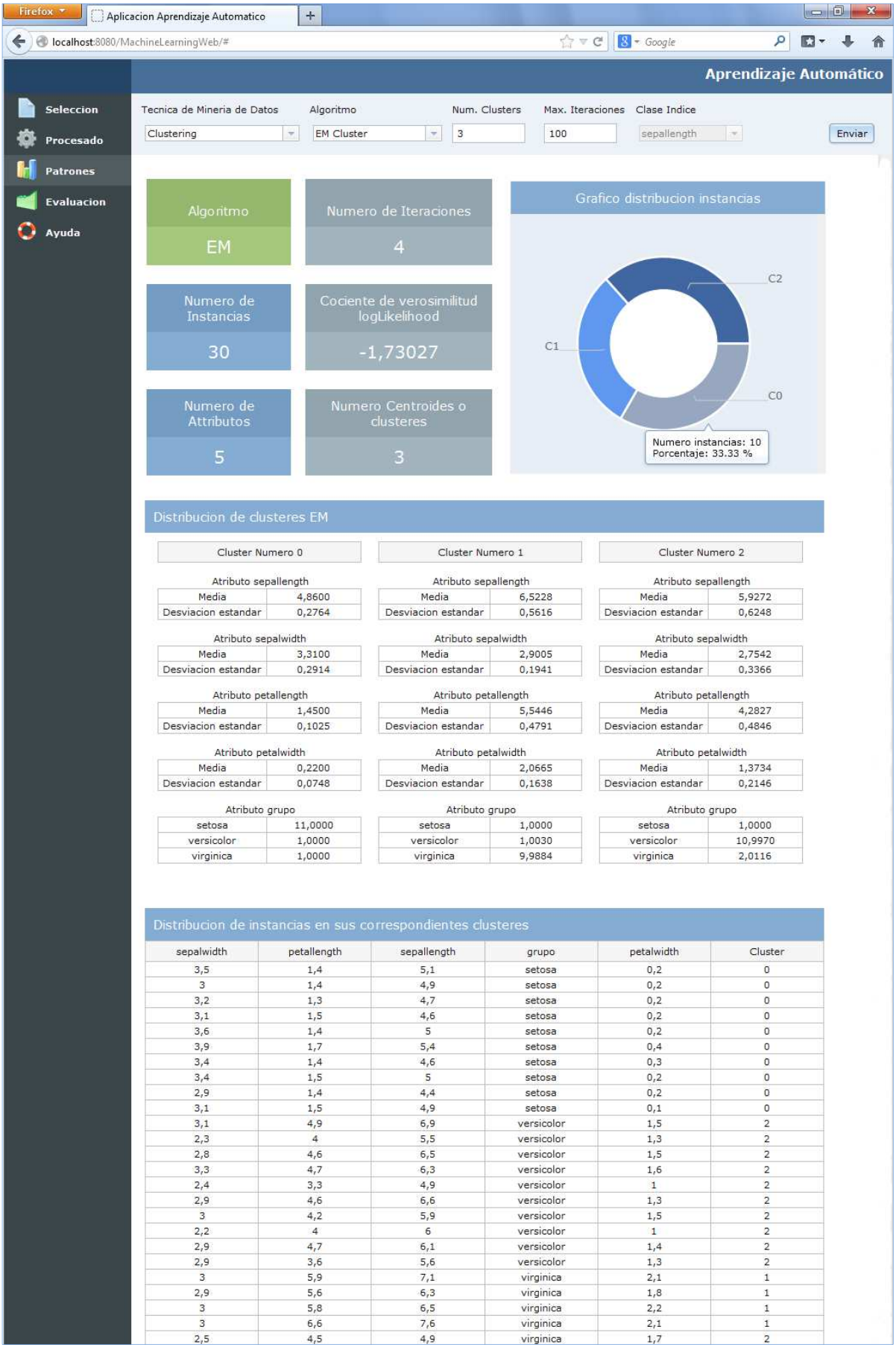


Figura 42 – Prueba entrenamiento algoritmo EM

5.3 Conjunto de pruebas de evaluación de algoritmos

En este apartado veremos las pruebas correspondientes a la evaluación de los distintos algoritmos de aprendizaje automático que soporta la aplicación, como su posterior presentación y visualización por parte del usuario. En total veremos dos pruebas descritas a continuación:

1. Prueba 7 - Evaluación del algoritmo EM
2. Prueba 8 - Evaluación del algoritmo KMENAS
3. Prueba 9 - Evaluación del algoritmo de árbol C4.5
4. Prueba 10 - Evaluación del algoritmo Naive Bayes

Nombre	**Prueba 7 - Evaluación del algoritmo EM**
Descripción	Evaluación del algoritmo EM
Entrada	Conjunto de datos y atributos debidamente procesado y transformado
Salida	Representación textual y grafica de los resultados obtenidos de la Evaluación del algoritmo EM
Requisitos	Tener un conjunto de datos y atributos cargado en la aplicación (conjunto de datos IRIS)
Proceso de Prueba	Cargamos un fichero de texto que contenga un conjunto de datos y atributos valido y lo procesamos con la aplicación para luego evaluar dichos datos con el algoritmo EM. La aplicación procesa dicha información y devuelve una representación textual y grafica del resultado de la evaluación de dicho algoritmo usando el conjunto de datos cargado en la aplicación.
Resultado esperado	Evalúa el algoritmo cambiando el número de clústeres de entrada de 1 a 10 y compara los resultados - Grafico que relaciona el número de clústeres con el logLikelihood (coeficiente de verosimilitud) - Tabla de evaluación que describe el grafico anterior. - Distribución de las instancias en sus correspondientes clústeres según el número total de clústeres de entrada.
Resultado obtenido	El resultado obtenido coincide con el resultado esperado

Veredicto	Pasa la prueba

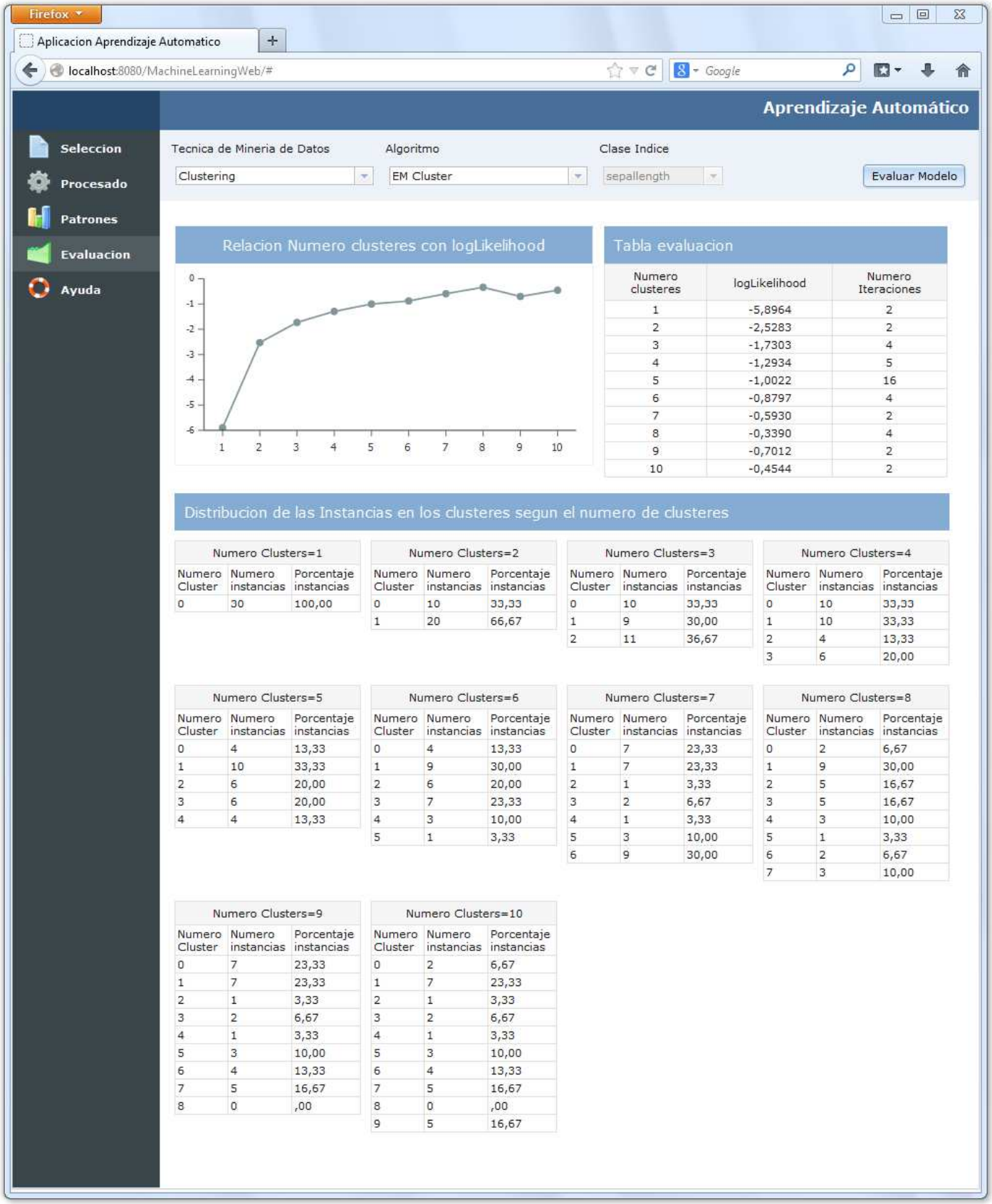
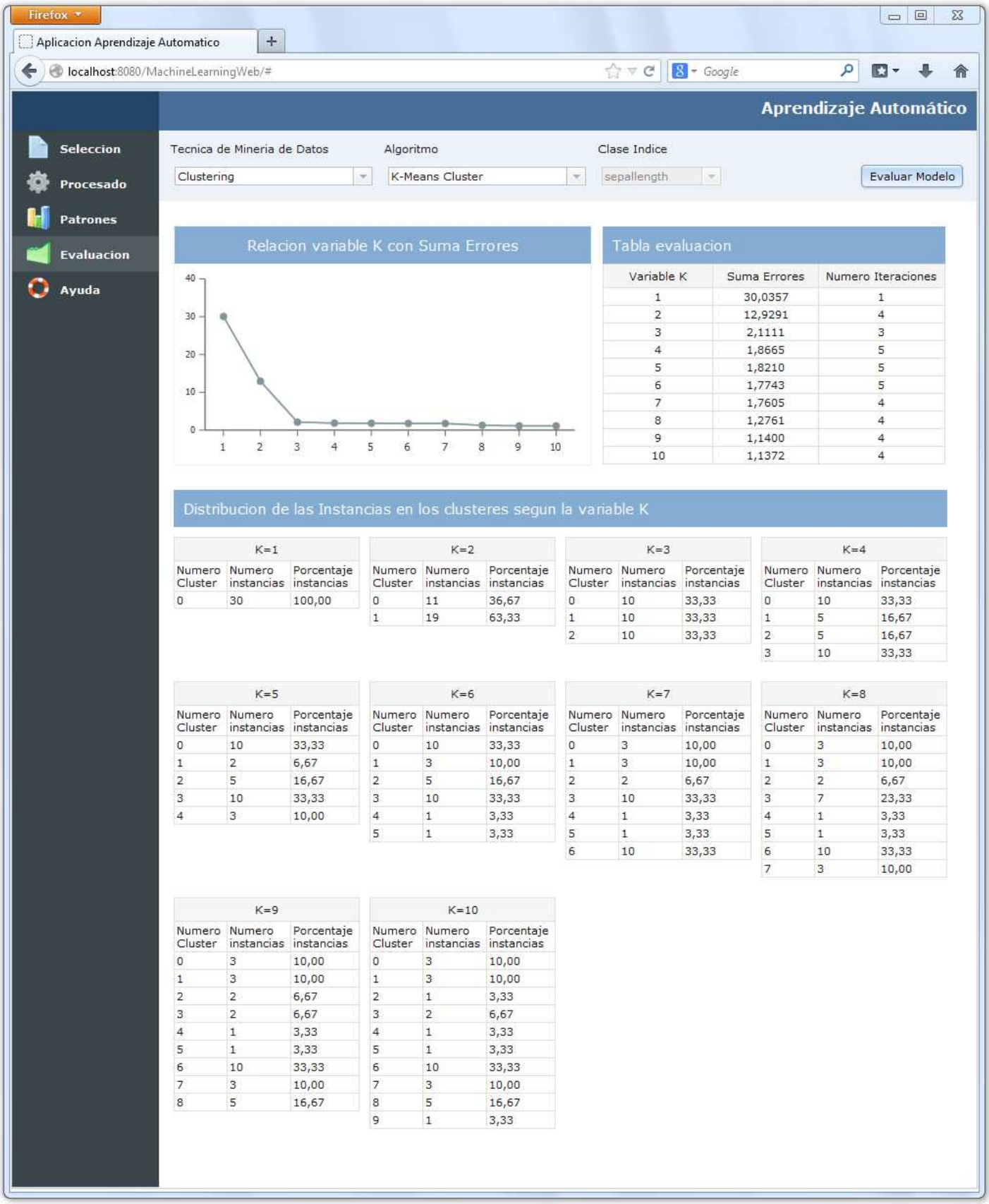


Figura 43 – Prueba evaluación algoritmo EM

Nombre	Prueba 8 - Evaluación del algoritmo K-Means
Descripción	Evaluación del algoritmo K-Means
Entrada	Conjunto de datos y atributos debidamente procesado y transformado
Salida	Representación textual y grafica de los resultados obtenidos de la Evaluación del algoritmo K-Means.
Requisitos	Tener un conjunto de datos y atributos cargado en la aplicación (Conjunto de datos IRIS).
Proceso de Prueba	Cargamos un fichero de texto que contenga un conjunto de datos y atributos valido y lo procesamos con la aplicación para luego evaluar dichos datos con el algoritmo K-Means, La aplicación procesa dicha información y devuelve una representación textual y grafica del resultado de la evaluación de dicho algoritmo usando el conjunto de datos cargado en la aplicación.
Resultado esperado	Evalúa el algoritmo cambiando la variable k (el número de clústeres de entrada) de 1 a 10 y compara los resultados. - Grafico que relaciona el número de clústeres de entrada (variable K) con la suma de errores cuadráticos. - Tabla de evaluación que describe el grafico anterior. - Distribución de las instancias en sus correspondientes clústeres según la variable k (número de clústeres de entrada).
Resultado obtenido	El resultado obtenido coincide con el resultado esperado.
Veredicto	Pasa la prueba.



Variable K	Suma Errores	Numero Iteraciones
1	30,0357	1
2	12,9291	4
3	2,1111	3
4	1,8665	5
5	1,8210	5
6	1,7743	5
7	1,7605	4
8	1,2761	4
9	1,1400	4
10	1,1372	4

K=1

Numero Cluster	Numero instancias	Porcentaje instancias
0	30	100,00

K=2

Numero Cluster	Numero instancias	Porcentaje instancias
0	11	36,67
1	19	63,33

K=3

Numero Cluster	Numero instancias	Porcentaje instancias
0	10	33,33
1	10	33,33
2	10	33,33

K=4

Numero Cluster	Numero instancias	Porcentaje instancias
0	10	33,33
1	5	16,67
2	5	16,67
3	10	33,33

K=5

Numero Cluster	Numero instancias	Porcentaje instancias
0	10	33,33
1	2	6,67
2	5	16,67
3	10	33,33
4	3	10,00

K=6

Numero Cluster	Numero instancias	Porcentaje instancias
0	10	33,33
1	3	10,00
2	5	16,67
3	10	33,33
4	1	3,33
5	1	3,33

K=7

Numero Cluster	Numero instancias	Porcentaje instancias
0	3	10,00
1	3	10,00
2	2	6,67
3	10	33,33
4	1	3,33
5	1	3,33
6	10	33,33

K=8

Numero Cluster	Numero instancias	Porcentaje instancias
0	3	10,00
1	3	10,00
2	2	6,67
3	7	23,33
4	1	3,33
5	1	3,33
6	10	33,33
7	3	10,00

K=9

Numero Cluster	Numero instancias	Porcentaje instancias
0	3	10,00
1	3	10,00
2	2	6,67
3	2	6,67
4	1	3,33
5	1	3,33
6	10	33,33
7	3	10,00
8	5	16,67

K=10

Numero Cluster	Numero instancias	Porcentaje instancias
0	3	10,00
1	3	10,00
2	1	3,33
3	2	6,67
4	1	3,33
5	1	3,33
6	10	33,33
7	3	10,00
8	5	16,67
9	1	3,33

Figura 44 – Prueba evaluación algoritmo K-Means

Nombre	Prueba 9 - Evaluación del algoritmo de árbol C4.5
Descripción	Evaluación del algoritmo de árbol C4.5
Entrada	Conjunto de datos y atributos debidamente procesado y transformado
Salida	Representación textual de los resultados obtenidos de la Evaluación del algoritmo de árbol C4.5
Requisitos	Tener un conjunto de datos y atributos cargado en la aplicación (conjunto de datos weather)
Proceso de Prueba	Cargamos un fichero que contenga un conjunto de datos y atributos valido y lo procesamos para luego evaluar dichos datos con el algoritmo de árbol C4.5, Como variable de entrada usaremos la clase índice o discriminatoria, La aplicación procesa dicha información y devuelve el resultado de la evaluación del algoritmo de árbol C4.5.
Resultado esperado	Coeficiente Kappa Precisión del algoritmo Instancias clasificadas correctamente y su porcentaje Instancias clasificadas incorrectamente y su porcentaje Matriz de confusión
Resultado obtenido	El resultado obtenido coincide con el resultado esperado
Veredicto	Pasa la prueba

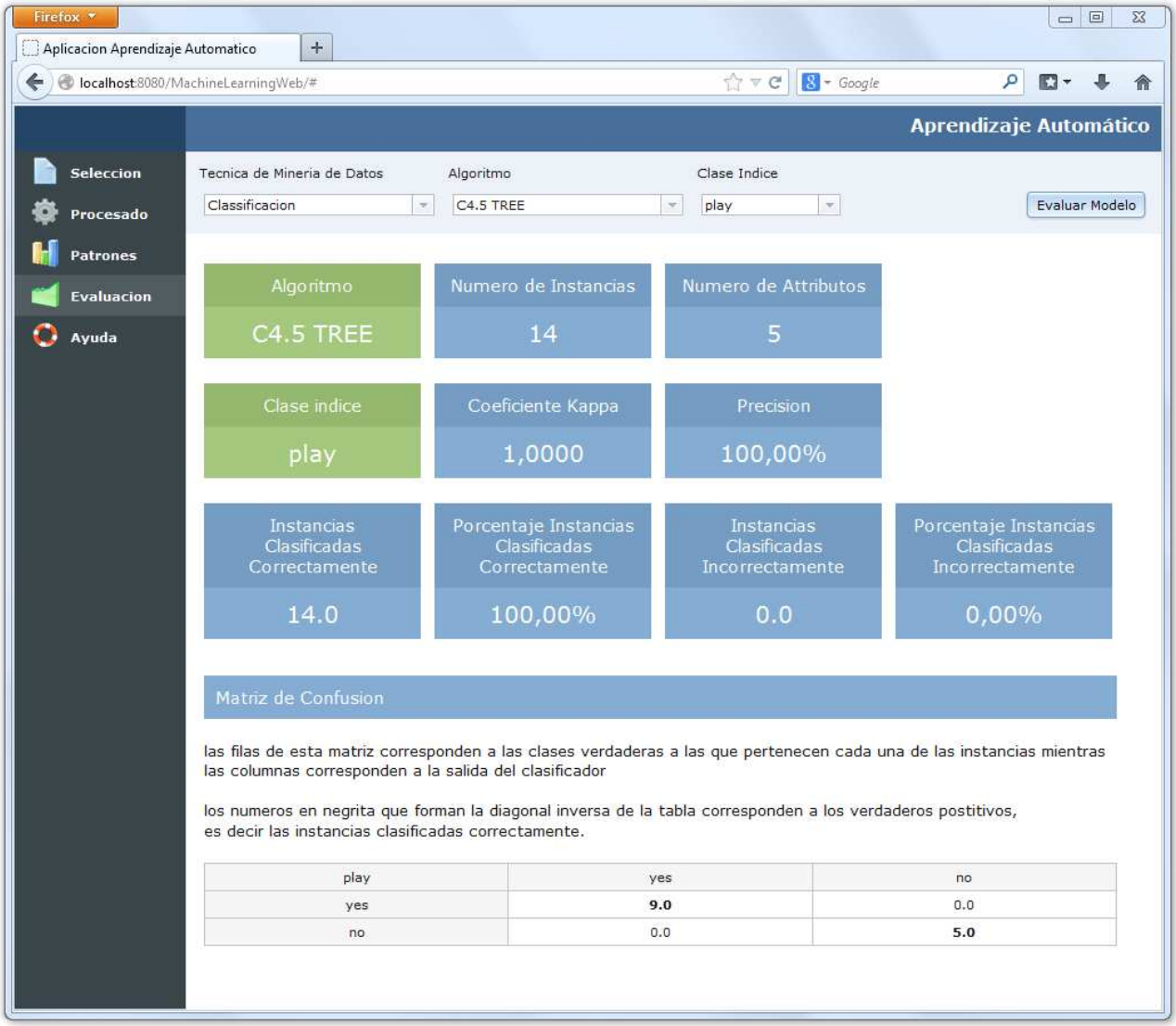


Figura 45 – Prueba evaluación algoritmo C4.5

Nombre	Prueba 10 - Evaluación del algoritmo Naive Bayes
Descripción	Evaluación del algoritmo Naive Bayes
Entrada	Conjunto de datos y atributos debidamente procesado y transformado
Salida	Representación textual de los resultados obtenidos de la Evaluación del algoritmo Naive Bayes
Requisitos	Tener un conjunto de datos y atributos cargado en la aplicación (conjunto de datos weather)
Proceso de Prueba	Cargamos un fichero que contenga un conjunto de datos y atributos valido y lo procesamos para luego evaluar dichos datos con el algoritmo Naive Bayes, Como variable de entrada usaremos la clase índice o discriminatoria. La aplicación procesa dicha información y devuelve resultado de la evaluación del algoritmo Naive Bayes.
Resultado esperado	Coeficiente Kappa Precisión del algoritmo Instancias clasificadas correctamente y su porcentaje Instancias clasificadas incorrectamente y su porcentaje Matriz de confusión
Resultado obtenido	El resultado obtenido coincide con el resultado esperado
Veredicto	Pasa la prueba

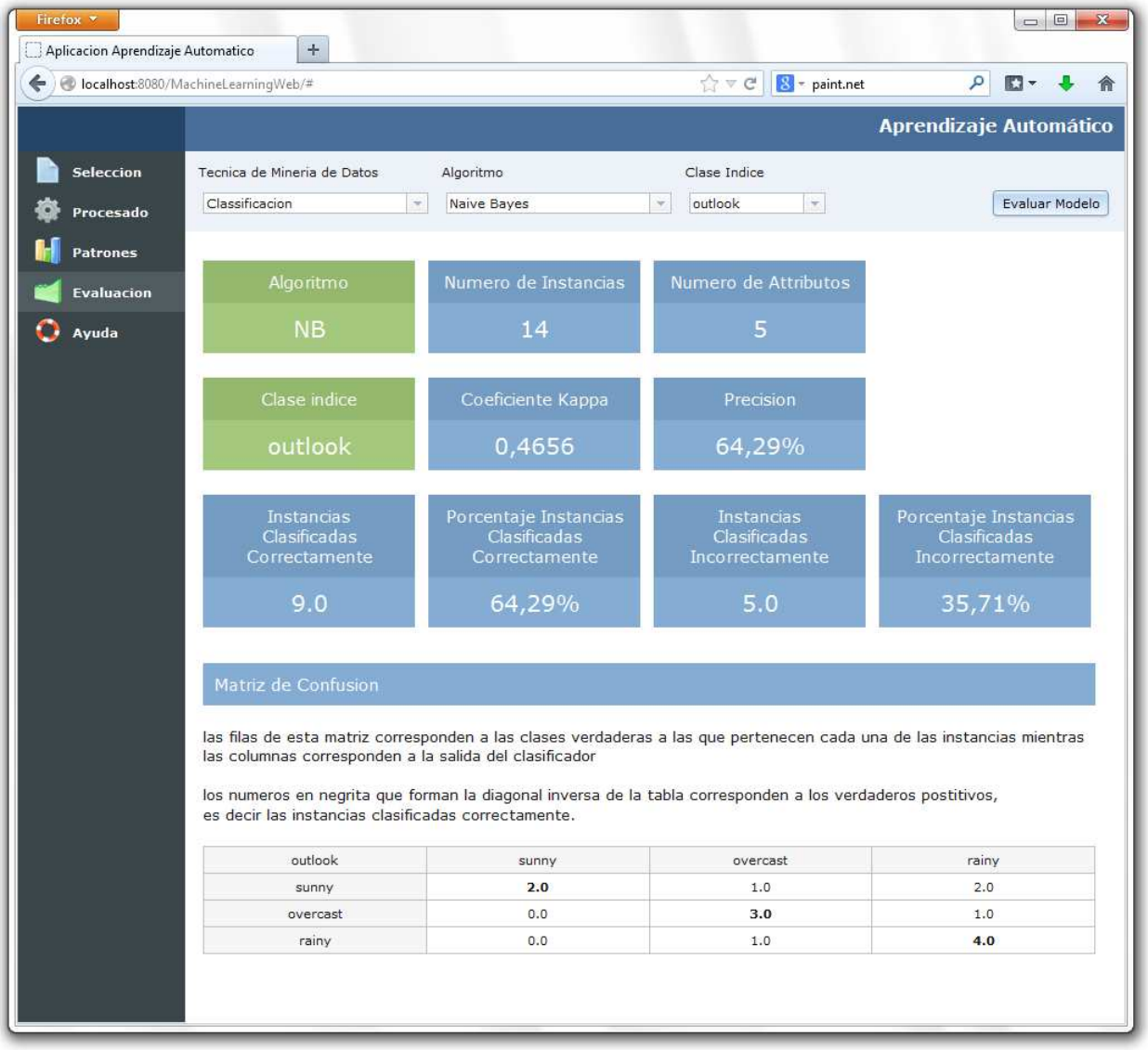


Figura 46 – Prueba evaluación algoritmo Naive Bayes

Capítulo 6

6. Conclusiones y futuras líneas de investigación

6.1. Conclusiones

Con este trabajo hemos presentado un escenario en el cual se presentan diversas técnicas de análisis de datos, para obtener patrones y extraer información de conjuntos de datos aplicando la minería de datos usando técnicas predictivas y descriptivas.

Se han estudiado técnicas y algoritmos de aprendizaje automático tanto de clasificación (usando modelos predictivos de minería de datos) como de agrupamiento de datos (usando modelos descriptivos de minería de datos), presentando las ventajas y/o inconvenientes para cada uno de dichos algoritmos o técnicas, En concreto se hizo un estudio de los siguientes algoritmos:

- Algoritmos de clasificación o técnicas de aprendizaje supervisado

    1. **ID3** clasifica datos mediante la construcción de un árbol de decisión
    2. **Naive Bayes** un clasificador probabilístico fundamentado en el teorema de Bayes

3. **SVM** máquinas de soporte vectorial, una SVM construye un hiperplano en un espacio de dimensionalidad muy alta que puede ser utilizado en problemas de clasificación o regresión.

Algoritmos de agrupamiento o técnicas de aprendizaje no supervisado

1. **K-Means** técnica de agrupamiento en el que cada observación pertenece al grupo más cercano a la media.
2. **Algoritmo esperanza-maximización o algoritmo EM** se usa en estadística para encontrar estimadores de máxima verosimilitud.

Además se han realizado experimentos usando conjunto de datos artificiales y reales para hacer un estudio comparativo entre los algoritmos antes mencionados. Y para completar el escenario y el estudio de las diversas técnicas de minería de datos y aprendizaje automático se ha desarrollado una aplicación y arquitectura web para entrenar algoritmos y evaluarlos usando distintos conjuntos de datos.

Dicha aplicación combina todas las herramientas necesarias para una completa gestión de los datos, desde la adquisición de los mismos y su posterior procesamiento para determinar las propiedades de sus atributos, hasta el entrenamiento y evaluación de dichos datos, ofreciendo al usuario diversos métodos de resolver un mismo problema para ello utiliza un conjunto de algoritmos tanto de agrupamiento como de clasificación.

Además ofrece al usuario una interfaz gráfica web intuitiva y completa la cual no solo facilita al usuario la adquisición y procesado de los datos, sino que también permite analizar los resultados del entrenamiento y evaluación e indagar en las características de los datos contenidos en los conjuntos de datos utilizados.

6.2. Futuras líneas de Investigación y mejoras

El campo del aprendizaje automático y minería de datos es un campo en pleno desarrollo, donde la mayoría de las herramientas utilizadas provienen de otros campos como el reconocimiento de patrones, la Estadística o la teoría de complejidad. Dada la novedad de las investigaciones en esta área quedan todavía varios problemas por afrontar, como ser el tamaño de los datos y el ruido en los mismos.

En los últimos años se han desarrollado muchos sistemas aprendizaje automático y se espera que este desarrollo continúe floreciendo dada la enorme cantidad de datos que son almacenados día a día, que requiere algún tipo de análisis o clasificación. La diversidad de los datos, y de las técnicas y enfoques de la minería de datos, son un desafío para el crecimiento de este área de la tecnología. Y en nuestro caso particular podemos plantear mejoras al presente proyecto, existen ciertos aspectos no considerados en este documento que podrían ser desarrollados como futuras líneas de investigación.

Debido al gran número de técnicas de aprendizaje automático, el escenario propuesto en este proyecto no se ha estudiado todas las técnicas y algoritmos existentes en la actualidad sino que solo se estudiaron las técnicas más representativas de cada caso y más utilizadas e importantes.

Con lo cual es posible ampliar el escenario a otras técnicas de aprendizaje automático y hacer un estudio sobre dichas técnicas de las cuales podríamos incluir las redes neuronales artificiales (RNA) las cuales son técnicas de aprendizaje y procesamiento automático inspirado en la forma en la que funciona el sistema nervioso de los animales, también podemos incluir Los algoritmos genéticos, los cuales son una técnica matemática de búsqueda y optimización que encuentra soluciones a un problema basándose en los principios que rigen la evolución de las especies a nivel genético molecular.

Además de lo mencionado anteriormente podemos añadir posibles mejoras en la aplicación desarrollada, podemos añadir una compatibilidad con mayor número de métodos o formas de adquisición o selección de los datos, actualmente solo hay un único método para alimentar la aplicación con conjuntos de datos y es a través de ficheros de texto en formato ARFF, añadir más métodos como por ejemplo obtener los datos a través de una base de datos u obtener datos a través de ficheros XML.

También es posible añadir mejoras en cuanto a la forma en la que se presentan los datos utilizando más gráficas, más tablas y más tipos de análisis de datos y presentar otros aspectos y características

tanto de los conjuntos de datos presentes en los documentos como los algoritmos aplicados sobre dichos datos.

La automatización de los procesos de evaluación de los algoritmos queda fuera del alcance de este proyecto como ya se menciono anteriormente en los objetivos del proyecto. Con lo cual una posible mejora seria automatizar la inclusión de algoritmos nuevos en la plataforma desarrollada para realizar los procesos de entrenamiento y evaluación.

En el apartado 4.10 se presento una guía para la inclusión de un nuevo algoritmo en la plataforma desarrollada, la idea sería automatizar el proceso que sigue dicha guía de tal forma que sea transparente para el usuario.

Dicho proceso no resultaría excesivamente difícil porque toda la plataforma se basa en módulos, los cuales funcionan de forma independiente a la implementación de los algoritmos. En el capítulo 4 se menciona que el paquete ConnectAPI hace de capa de transparencia o separación para que nuestra aplicación no dependa de la API de implementación de algoritmos. Dicho paquete se puede encapsular en un JAR y usarlo como si fuera una biblioteca mas.

Con lo que la idea sería desarrollar un paquete de clases similar a ConnectAPI y encapsularlo en un JAR por cada API de implementación de algoritmos que utilicemos en la aplicación.

Además de lo mencionado anteriormente habrá que adaptar la interfaz de usuario para que pueda hacer el cambio entre una API y otra de forma transparente para el usuario.

Bibliografía

[AGR93] R. Agrawal; T. Imielinski; A. Swami: Mining Association Rules Between Sets of Items in Large Databases",
SIGMOD Conference 1993: 207-216

[BAY63] Bayes, Thomas (1763). «An Essay towards solving a Problem in the Doctrine of Chances.».
Philosophical Transactions of the Royal Society of London 53: pp. 370–418. doi:10.1098/rstl.1763.0053.

[BRE84] Breiman, Leo; Friedman, J. H.; Olshen, R. A.; Stone, C. J. (1984).
Classification and regression trees. Monterey, CA: Wadsworth & Brooks/Cole Advanced Books & Software.
ISBN 978-0-412-04841-8.

[BRY63] Bryson, A.E.; W.F. Denham; S.E. Dreyfus.
Optimal programming problems with inequality constraints. I: Necessary conditions for extremal solutions. AIAA J. 1, 11 (1963) 2544-2550

[DEM77] Dempster, A.P.; Laird, N.M.; Rubin, D.B. (1977).
"Maximum Likelihood from Incomplete Data via the EM Algorithm".
Journal of the Royal Statistical Society, Series B 39 (1):
1–38. JSTOR 2984875. MR 0501537.

[DOM05] M. Domínguez-Dorado,. Todo Programación. Nº 12. Págs. 16-20. Editorial Iberprensa (Madrid).DL M-13679-2004. Septiembre, 2005.. Programación de algoritmos genéticos..

[dwreview2.6] http://www.dwreview.com/DW_Overview.html
ultimo acceso : 10 junio 2014

[DYC00] Jill Dyché (2000). e-Data: turning data into information with data warehousing. Addison-Wesley. p.323.
Alexander Factor (2001). Analyzing Application Service Providers. Prentice Hall PTR. p.290.

[EST96] Ester, Martin; Kriegel, Hans-Peter; Sander, Jörg; Xu, Xiaowei (1996).
"A density-based algorithm for discovering clusters in large spatial databases with noise". In Simoudis, Evangelos; Han, Jiawei; Fayyad, Usama M. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96). AAAI Press. pp. 226–231. ISBN 1-57735-004-9.CiteSeerX: 10.1.1.71.1980.

[FAY96] From data mining to knowledge discovery
Usama M. Fayyad,Gregory Piatetsky-Shapiro,Padhraic Smyth
Published in: Book Advances in knowledge discovery and data mining
Pages 1 – 34 American Association for Artificial Intelligence Menlo Park, CA, USA 1996 ISBN:0-262-56097-6

[FIS87] Fisher, Douglas (1987). "Knowledge acquisition via incremental conceptual clustering". Machine Learning 2 (2): 139–172. doi:10.1007/BF00114265.

[FIS87b] Fisher, Douglas H. (July 1987). "Improving inference through conceptual clustering".Proceedings of the 1987 AAAI Conferences. AAAI Conference. Seattle Washington. pp. 461–465.

[HAL11] Data Mining : Practical Machine Learning Tools and Techniques,
I. H.Witten, E. Frank & M. A. Hall, 3ª. Edición,
Morgan Kaufmann Publishers,Elsevier, USA, 2011.Págs285–288

[HOD89] Fix, E.; J.L. Hodges (1989). «(1951): An Important Contribution to Nonparametric Discriminant
Analysis and Density Estimation: Commentary on Fix and Hodges (1951)». International Statistical Review / Revue Internationale de Statistique 57 (3):
pp. 233-238. doi:doi:10.2307/1403796.

[KOH07] Kohonen, Teuvo; Honkela, Timo (2007). "Kohonen Network". Scholarpedia.

[KOH82] Kohonen, Teuvo (1982). "Self-Organized Formation of Topologically Correct Feature Maps".Biological Cybernetics 43 (1): 59–69. doi:10.1007/bf00337288.

[MAC67] MacQueen, J. B. (1967). "Some Methods for classification and Analysis of Multivariate Observations".Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability 1.University of California Press. pp. 281–297. MR 0214227. Zbl 0214.46201. Retrieved 2009-04-07.

[MEN03] T. Menzies, Y. Hu. Data Mining For Busy People. IEEE Computer, Outubro de 2003, pgs. 18-25.

[microsoft3.7.1] http://research.microsoft.com/en-us/um/people/manik/projects/trade-off/svm.html

[MIT97] Machine Learning, Tom Mitchell, McGraw Hill, 1997.
414 pages. ISBN 0070428077

[MON07] mineria de datos:tecnicas y heramientas
jose maria montero lorenzo
international thomson ediciones paraninfo s.a.1 edicion,2007

[netbeans4.3.1] https://netbeans.org/kb/docs/javaee/ecommerce/design.html

[ODE10] Oded Maimon and Lior Rokach (2010). Data Mining and Knowledge Discovery Handbook.
Springer, New Your. ISBN 978-0-387-09823-4.

[opencv3.7.1] http://docs.opencv.org/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html

[PAN06] Introduction to Data Mining, Pang-Ning Tan, Michigan State University, Michael Steinbach, University of Minnesota , Vipin Kumar, University of Minnesota (March 25, 2006) Addison-Wesley

[PIA91] Piatetsky-Shapiro, G. (1991), Discovery, analysis, and presentation of strong rules,
in G. Piatetsky-Shapiro & W. J. Frawley, eds, 'Knowledge Discovery in Databases', AAAI/MIT Press,Cambridge, MA.

[QUI86] Quinlan, J. R., (1986). Induction of Decision Trees. Machine Learning 1: 81-106, Kluwer Academic Publishers

[QUI93] Quinlan, J. R. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, 1993.

[ROS57] Rosenblatt, Frank (1957), The Perceptron--a perceiving and recognizing automaton.
Report 85-460-1, Cornell Aeronautical Laboratory.

[SKYCAT2.9.3] http://www.worldskycat.com/skycat/skykitten.html
ultimo acceso : 10 junio 2014

[SOM09] Machine Learning with SVM and Other Kernel Methods
Escrito por K.P. Soman,R. LOGANATHAN,V. AJAY
2009 - PHI Learning Private Limited
pag 122

[SPI07] M.R. Spiegel; J. Schiller; R. A. Srinivasan (2007).
«9. Análisis de la varianza». Probabilidad y Estadística
[Schaum's Outline of Theory and Problems of Probability and Statistics].
Schaum (2ª edición). México D.F.: McGraw-Hill. pp. 335–371. ISBN 978-970-10-4231-1.

[Spring4.3.1] http://docs.spring.io/spring/docs/1.2.9/reference/introduction.html

[STE57] Steinhaus, H. (1957). "Sur la division des corps matériels en parties".
Bull. Acad. Polon. Sci. (in French) 4 (12): 801–804. MR 0090073. Zbl 0079.16403.

[TUK83] Hoaglin, D C; Mosteller, F & Tukey, John Wilder (Eds) (1983).
Understanding Robust and Exploratory Data Analysis.
ISBN 0-471-09777-2.

[weka5] http://www.cs.waikato.ac.nz/ml/weka/datasets.html

[wikibooks2.4.2] http://en.wikibooks.org/wiki/Data_Mining_Algorithms_In_R/Clustering/K-Means
ultimo acceso : 10 junio 2014

[wikipedia3] http://es.wikipedia.org/wiki/Aprendizaje_automatico

[wikipedia2.4.1] http://es.wikipedia.org/wiki/Red_neuronal_artificial
ultimo acceso : 15 junio 2014

[wikipedia4.2] http://en.wikipedia.org/wiki/Model_2
ultimo acceso : 15 junio 2014

[wikipedia4.7] http://es.wikipedia.org/wiki/Caso_de_uso
ultimo acceso : 15 junio 2014

[XIN07] Xingquan Zhu, Ian Davidson (2007). Knowledge Discovery and Data Mining: Challenges and Realities. Hershey, New Your. p. 18.
ISBN 978-1-59904-252-7.

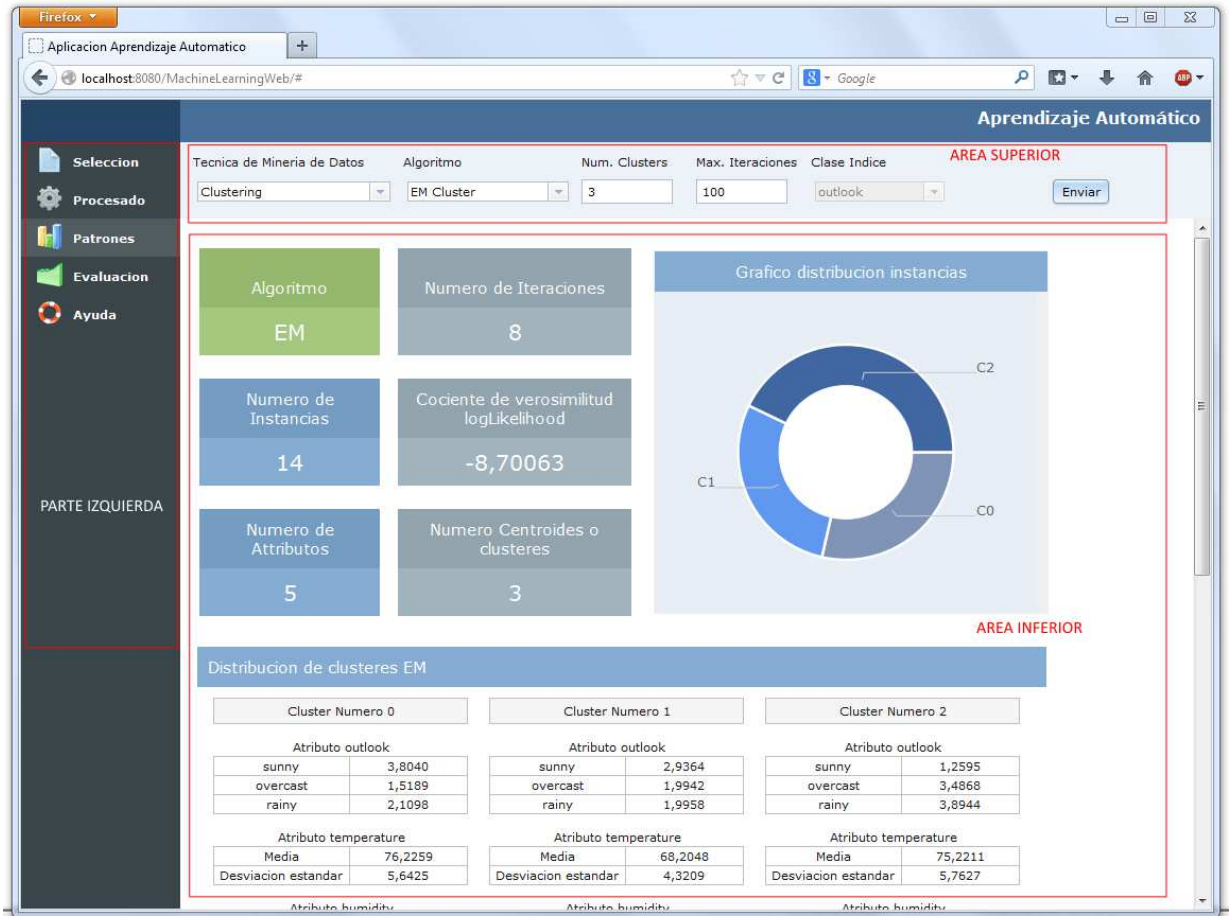
Anexo I - Guía de usuario

Este anexo es una pequeña guía de usuario para mostrar las principales funcionalidades de la aplicación desarrollada, a continuación explicamos las distintas ventanas que componen la aplicación:

La ventana principal contiene dos áreas, una parte izquierda donde tenemos el menú principal, que contiene enlaces a las distintas ventanas de la aplicación, este menú es fijo y siempre estará a la vista para permitir cambiar entre las ventanas con facilidad.

Y una parte derecha que ira cambiando según en qué ventana nos encontremos, que a su vez contiene dos áreas:

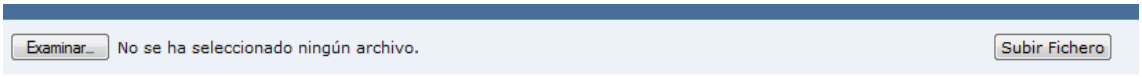
- Un área superior donde encontramos los distintos controles para subir un fichero, procesar un atributo, entrenar un algoritmo, evaluar un algoritmo, etc.
- Un área inferior donde se presenta el resultado de las operaciones realizadas sobre el conjunto de datos



1. Ventana Selección

1.1 Área superior

En el área superior tenemos un formulario que nos permite subir ficheros de datos que contienen un conjunto de datos y cargarlos en la aplicación, a continuación vemos la figura correspondiente a esa área

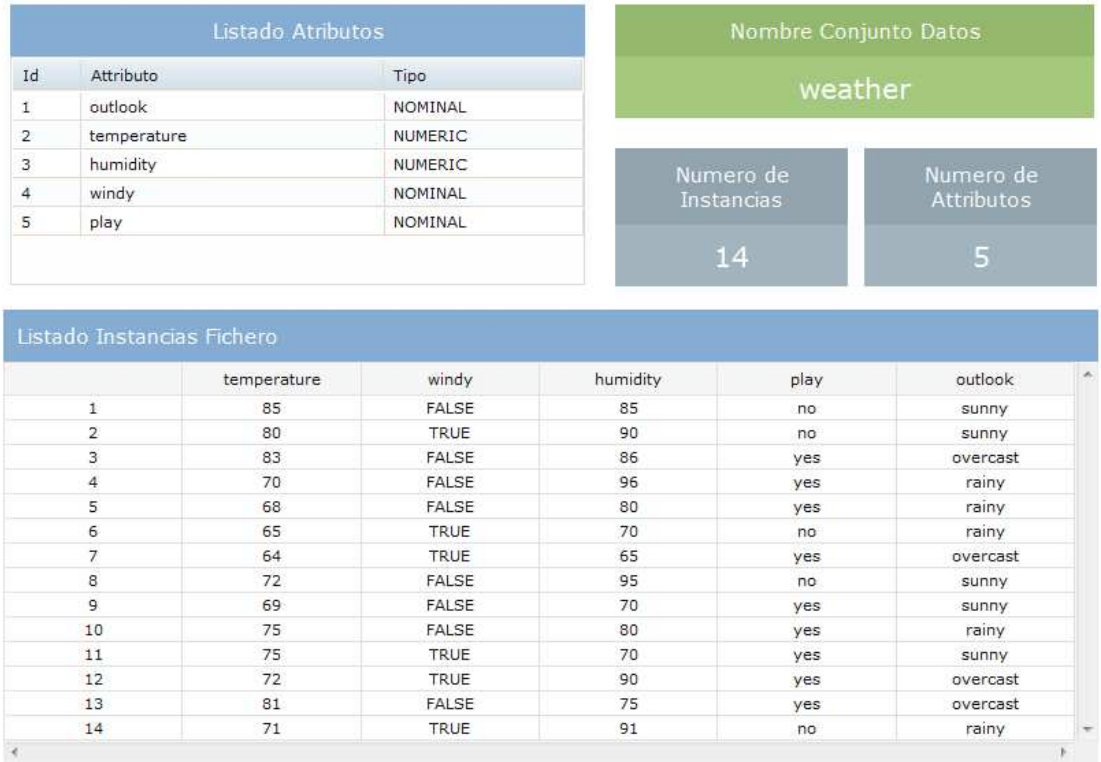


1.2 Área Inferior

En el área inferior se presentan los datos que se han cargado en la aplicación, se puede visualizar en esa área los siguientes datos

- Un listado de atributos con su nombre y tipo.
- El nombre del conjunto de datos cargado.
- El número de instancias que contiene el fichero.
- El número de atributos que pertenecen al conjunto de datos cargado.
- Una tabla con todas las instancias del fichero cargado en la aplicación.

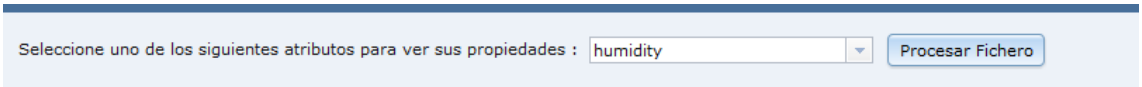
A continuación vemos la figura correspondiente a esa área



2. Ventana Procesado

2.1 Área superior

En el área superior tenemos un formulario que contiene un selector con todos los atributos del conjunto de datos cargado en la aplicación, seleccionando cualquier atributo podemos visualizar las propiedades de dicho atributo en el área inferior a continuación vemos la figura correspondiente a esa área

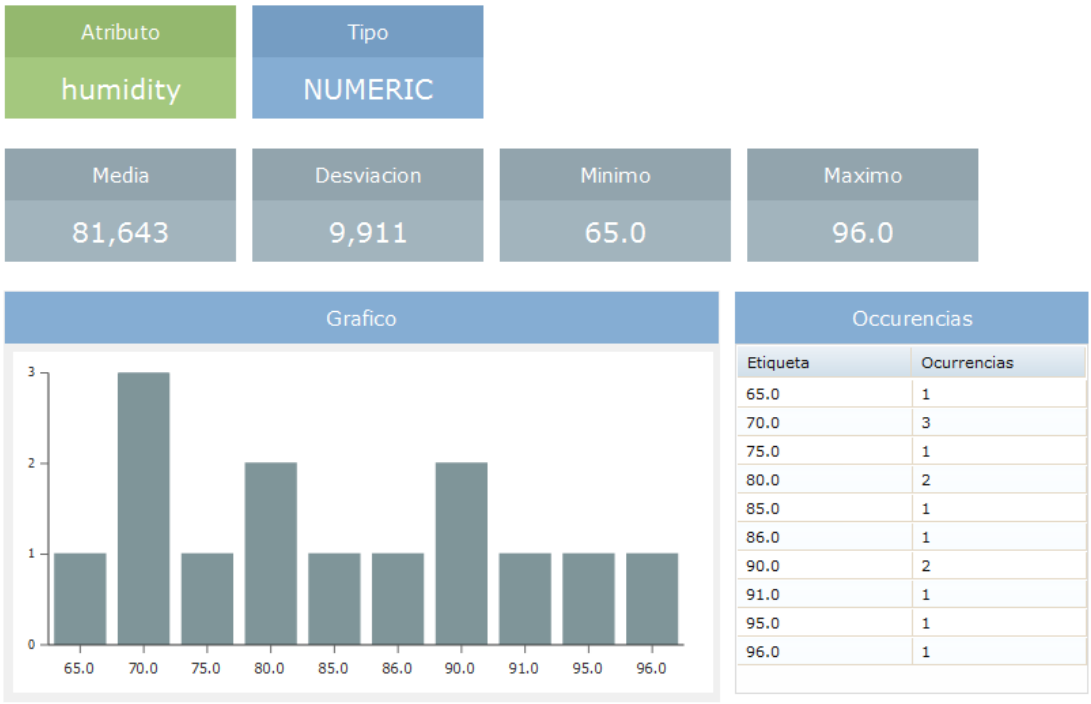


2.2 Área inferior

En el área inferior se presentan las propiedades del atributo seleccionado, se puede visualizar en esa área los siguientes datos

- Nombre del atributo seleccionado.
- Tipo del atributo seleccionado (numérico o nominal).
- Un listado de las ocurrencias de todos los valores que puede tomar el atributo seleccionado.
- Un gráfico que muestra gráficamente el listado de las ocurrencias anterior.
- Y solo para los atributos numéricos veremos también la media, desviación estándar, mínimo y máximo.

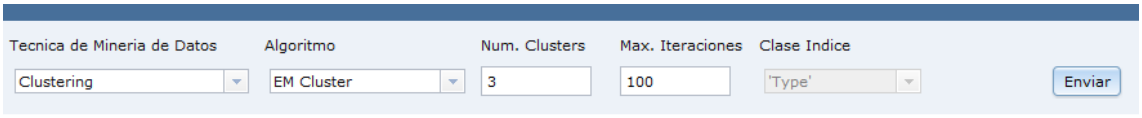
A continuación vemos la figura correspondiente a esa área



3. Ventana Patrones

3.1 Área superior

En el área superior tenemos un formulario que contiene varios selectores que nos permiten entrenar un conjunto de datos usando una técnica de minería de datos y el correspondiente algoritmo de aprendizaje automático así como otros controles para modificar variables de inicio como el número de clústeres, número máximo de iteraciones y la clase índice, a continuación vemos la figura correspondiente a esa área

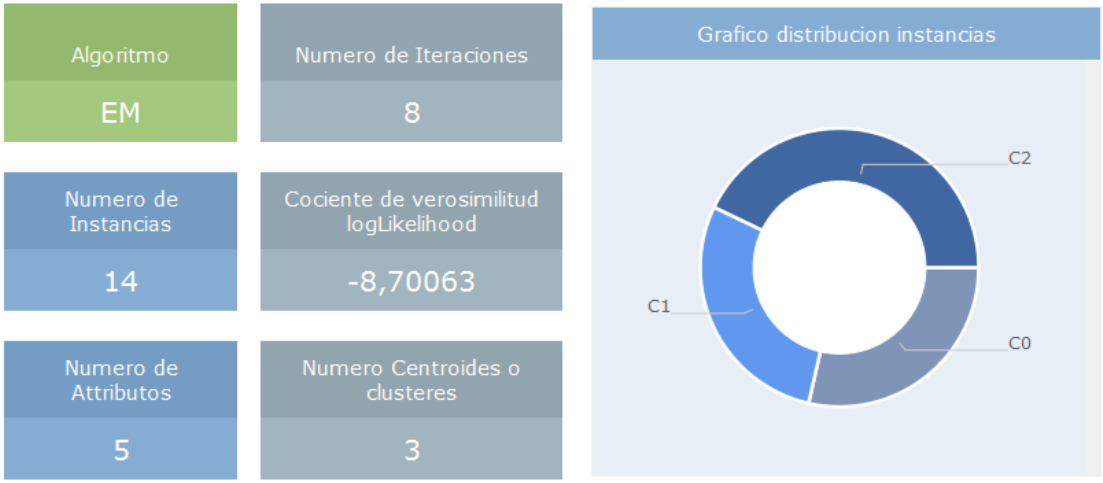


3.2 Área Inferior

En el área inferior se muestra el resultado del entrenamiento del conjunto de datos con el algoritmo seleccionado, los datos visualizados cambian según el algoritmo y la técnica seleccionados

Esa área a su vez se divide en tres áreas:

1. La primera área contiene datos genéricos del algoritmo y conjunto de datos y un gráfico de distribución de instancias en sus correspondientes clústeres o clases.



2. Una segunda área que varía según el algoritmo seleccionado

Para el algoritmo EM muestra una distribución de todos los clústeres con sus correspondientes atributos y las propiedades de cada uno de estos

Distribucion de clusteres EM

Cluster Numero 0	
Atributo outlook	
sunny	3,8040
overcast	1,5189
rainy	2,1098
Atributo temperature	
Media	76,2259
Desviacion estandar	5,6425
Atributo humidity	
Media	90,4605
Desviacion estandar	3,4609
Atributo windy	
TRUE	3,3652
FALSE	3,0675
Atributo play	
yes	1,6899
no	4,7428

Cluster Numero 1	
Atributo outlook	
sunny	2,9364
overcast	1,9942
rainy	1,9958
Atributo temperature	
Media	68,2048
Desviacion estandar	4,3209
Atributo humidity	
Media	68,7352
Desviacion estandar	2,1750
Atributo windy	
TRUE	3,9602
FALSE	1,9663
Atributo play	
yes	3,9307
no	1,9958

Cluster Numero 2	
Atributo outlook	
sunny	1,2595
overcast	3,4868
rainy	3,8944
Atributo temperature	
Media	75,2211
Desviacion estandar	5,7627
Atributo humidity	
Media	83,6985
Desviacion estandar	7,0565
Atributo windy	
TRUE	1,6746
FALSE	5,9662
Atributo play	
yes	6,3794
no	1,2614

Para el algoritmo K-Means muestra un listado de los centriodes obtenidos al entrenar el conjunto de datos cargado en la aplicación.

Centroides K-Means

temperature	windy	humidity	play	outlook	Cluster
74,3333	FALSE	81,1667	yes	rainy	0
70,3333	TRUE	75	yes	overcast	1
74,6	TRUE	86,2	no	sunny	2

Para el algoritmo C4.5 muestra una figura de un árbol que representa la distribución de las clases del conjunto de datos.

Distribucion de Clases en Arbol ID3

Y finalmente para el algoritmo Naive Bayes muestra una distribución de todas las clases con sus correspondientes atributos y las propiedades de cada uno de estos

Distribucion de Clases Naive Bayes

Clase Numero 0 Probabilidad : 0,3529 Etiqueta :sunny	
Atributo temperature	
Media	76,3636
Desviacion estandar	6,0371
Atributo humidity	
Media	81,9778
Desviacion estandar	11,1931
Atributo windy	
TRUE	3,0000
FALSE	4,0000
Atributo play	
yes	3,0000
no	4,0000

Clase Numero 1 Probabilidad : 0,2941 Etiqueta :overcast	
Atributo temperature	
Media	74,9318
Desviacion estandar	6,8001
Atributo humidity	
Media	79,2222
Desviacion estandar	9,4330
Atributo windy	
TRUE	3,0000
FALSE	3,0000
Atributo play	
yes	5,0000
no	1,0000

Clase Numero 2 Probabilidad : 0,3529 Etiqueta :rainy	
Atributo temperature	
Media	69,8727
Desviacion estandar	3,1019
Atributo humidity	
Media	82,6667
Desviacion estandar	9,4957
Atributo windy	
TRUE	3,0000
FALSE	4,0000
Atributo play	
yes	4,0000
no	3,0000

3. Una tercera área común que contiene un listado de todas las instancias del conjunto de datos, asociadas cada una de ellas a su correspondiente clúster o clase

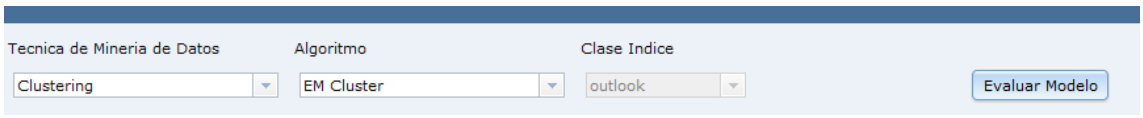
Distribucion de instancias en sus correspondientes clusteres

temperature	windy	humidity	play	outlook	Cluster
85	FALSE	85	no	sunny	0
80	TRUE	90	no	sunny	0
83	FALSE	86	yes	overcast	2
70	FALSE	96	yes	rainy	2
68	FALSE	80	yes	rainy	2
65	TRUE	70	no	rainy	1
64	TRUE	65	yes	overcast	1
72	FALSE	95	no	sunny	0
69	FALSE	70	yes	sunny	1
75	FALSE	80	yes	rainy	2
75	TRUE	70	yes	sunny	1
72	TRUE	90	yes	overcast	2
81	FALSE	75	yes	overcast	2
71	TRUE	91	no	rainy	0

4. Ventana Evaluación

4.1 Área superior

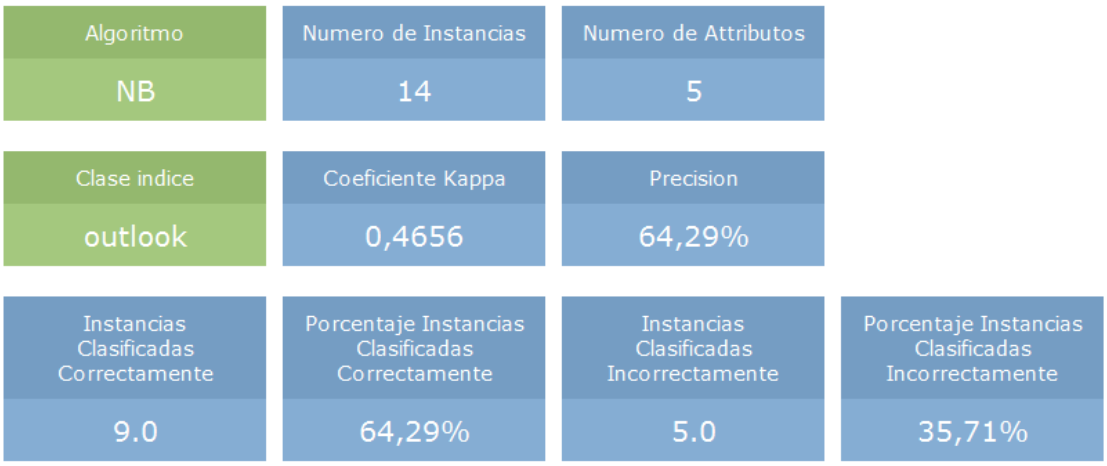
En el área superior tenemos un formulario que contiene varios selectores que nos permiten evaluar una técnica de minería de datos y el correspondiente algoritmo de aprendizaje automático, así como otros controles para modificar variables de inicio como la clase índice, a continuación vemos la figura correspondiente a esa área



4.2 Área Inferior

En el área inferior se muestra el resultado de la evaluación del algoritmo seleccionado los datos visualizados cambian según el algoritmo y la técnica seleccionados,

Para los algoritmos de clasificación naive bayes y c4.5 la ventana se divide en dos áreas, una superior donde se muestran datos genéricos de la evaluación del algoritmo como porcentajes y precisión, coeficiente kappa, etc...



Y más abajo la matriz de confusión correspondiente a la evaluación del algoritmo seleccionado.

Matriz de Confusion

las filas de esta matriz corresponden a las clases verdaderas a las que pertenecen cada una de las instancias mientras las columnas corresponden a la salida del clasificador

los numeros en negrita que forman la diagonal inversa de la tabla corresponden a los verdaderos postitivos, es decir las instancias clasificadas correctamente.

outlook	sunny	overcast	rainy
sunny	**2.0**	1.0	2.0
overcast	0.0	**3.0**	1.0
rainy	0.0	1.0	**4.0**

Para el algoritmo K-Means nos muestra los siguientes datos

Una tabla que representa una relación entre la variable K y la suma de errores cuadráticos y número de iteraciones y a su izquierda una representación gráfica de los datos de esta tabla.

Relacion variable K con Suma Errores

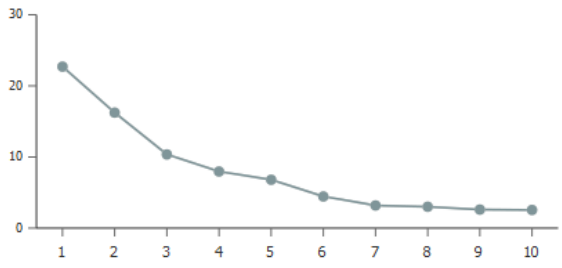


Tabla evaluacion

Variable K	Suma Errores	Numero Iteraciones
1	22,7041	1
2	16,2375	3
3	10,3657	3
4	7,9644	3
5	6,8050	4
6	4,4550	2
7	3,1924	2
8	3,0114	2
9	2,6137	2
10	2,5581	2

Y más abajo nos muestra la distribución de las instancias en sus correspondientes clústeres para valores de la variable K que van desde 1 hasta el 10

Distribucion de las Instancias en los clusteres segun la variable K

K=1

Numero Cluster	Numero instancias	Porcentaje instancias
0	14	100,00

K=2

Numero Cluster	Numero instancias	Porcentaje instancias
0	9	64,29
1	5	35,71

K=3

Numero Cluster	Numero instancias	Porcentaje instancias
0	6	42,86
1	3	21,43
2	5	35,71

K=4

Numero Cluster	Numero instancias	Porcentaje instancias
0	4	28,57
1	3	21,43
2	5	35,71
3	2	14,29

K=5

Numero Cluster	Numero instancias	Porcentaje instancias
0	3	21,43
1	2	14,29
2	4	28,57
3	3	21,43
4	2	14,29

K=6

Numero Cluster	Numero instancias	Porcentaje instancias
0	4	28,57
1	3	21,43
2	2	14,29
3	1	7,14
4	1	7,14
5	3	21,43

K=7

Numero Cluster	Numero instancias	Porcentaje instancias
0	3	21,43
1	2	14,29
2	2	14,29
3	1	7,14
4	1	7,14
5	3	21,43
6	2	14,29

K=8

Numero Cluster	Numero instancias	Porcentaje instancias
0	2	14,29
1	2	14,29
2	2	14,29
3	1	7,14
4	1	7,14
5	3	21,43
6	2	14,29
7	1	7,14

K=9

Numero Cluster	Numero instancias	Porcentaje instancias
0	2	14,29
1	1	7,14
2	2	14,29
3	1	7,14
4	1	7,14
5	3	21,43
6	2	14,29
7	1	7,14
8	1	7,14

K=10

Numero Cluster	Numero instancias	Porcentaje instancias
0	1	7,14
1	1	7,14
2	2	14,29
3	1	7,14
4	1	7,14
5	3	21,43
6	2	14,29
7	1	7,14
8	1	7,14
9	1	7,14

Para el algoritmo EM nos muestra los siguientes datos:

Una tabla que representa una relación entre el número de clústeres y el loglikelihood y número de iteraciones y a su izquierda una representación gráfica de los datos de esta tabla.

Relacion Numero clusteres con logLikelihood

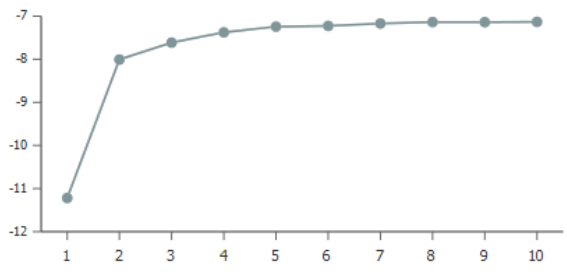


Tabla evaluacion

Numero clusteres	logLikelihood	Numero Iteraciones
1	-11,2188	1
2	-8,0071	9
3	-7,6147	23
4	-7,3772	56
5	-7,2432	15
6	-7,2267	18
7	-7,1701	16
8	-7,1371	23
9	-7,1391	22
10	-7,1327	10

Y más abajo nos muestra la distribución de las instancias en sus correspondientes clústeres para valores de clúster entre 1 y 10, es decir empieza evaluando el algoritmo seleccionado asignando todas las instancias a un único clúster, después asignando las instancias a 2 clústeres y así hasta evaluar algoritmo seleccionado asignando las instancias a 10 clústeres.

Distribucion de las Instancias en los clusteres segun el numero de clusteres

Numero Clusters=1

Numero Cluster	Numero instancias	Porcentaje instancias
0	435	100,00

Numero Clusters=2

Numero Cluster	Numero instancias	Porcentaje instancias
0	237	54,48
1	198	45,52

Numero Clusters=3

Numero Cluster	Numero instancias	Porcentaje instancias
0	78	17,93
1	151	34,71
2	206	47,36

Numero Clusters=4

Numero Cluster	Numero instancias	Porcentaje instancias
0	40	9,20
1	202	46,44
2	130	29,89
3	63	14,48

Numero Clusters=5

Numero Cluster	Numero instancias	Porcentaje instancias
0	80	18,39
1	134	30,80
2	128	29,43
3	53	12,18
4	40	9,20

Numero Clusters=6

Numero Cluster	Numero instancias	Porcentaje instancias
0	73	16,78
1	100	22,99
2	128	29,43
3	52	11,95
4	40	9,20
5	42	9,66

Numero Clusters=7

Numero Cluster	Numero instancias	Porcentaje instancias
0	56	12,87
1	36	8,28
2	136	31,26
3	127	29,20
4	36	8,28
5	27	6,21
6	17	3,91

Numero Clusters=8

Numero Cluster	Numero instancias	Porcentaje instancias
0	30	6,90
1	34	7,82
2	14	3,22
3	127	29,20
4	36	8,28
5	34	7,82
6	12	2,76
7	148	34,02

Numero Clusters=9

Numero Cluster	Numero instancias	Porcentaje instancias
0	35	8,05
1	38	8,74
2	38	8,74
3	116	26,67
4	28	6,44
5	106	24,37
6	15	3,45
7	39	8,97
8	20	4,60

Numero Clusters=10

Numero Cluster	Numero instancias	Porcentaje instancias
0	26	5,98
1	46	10,57
2	51	11,72
3	112	25,75
4	9	2,07
5	101	23,22
6	25	5,75
7	41	9,43
8	20	4,60
9	4	,92

ANEXO II - Código fuente

Paquete:com.machine.learning.bean

Clase:AtributoBean.java

```
package com.machine.learning.bean;
import java.util.LinkedHashMap;

public class AtributoBean {

 /** nombre del atributo*/
 private String nombre;
 /** tipo del atributo , nominal o numerico*/
 private String tipo;
 /** valor del atributo*/
 private String valor;
 /** valor minimo del atributo */
 private double min;
 /** valor maximo del atributo */
 private double max;
 /** valor medio del atributo */
 private double media;
 /** desviacion estandar del valor del atributo */
 private double stdDev;
 /** hashmap que contiene los valores que puede tomar
  * un atributo y el numero de ocurrencias de cada valor en
  * el conjunto de datos*/
 private LinkedHashMap labels;
 /** array que contiene los posibles valores que puede tomar un atributo*/
 private String[] posiblesValores;

 /** Constructor por defecto */
 public AtributoBean() {
       super();
 }

 /** Constructor
  * @param nombre : nombre del atributo
```

```
 * @param tipo     : tipo del atributo
 * @param valor  : valor del atributo*/
public AtributoBean(String nombre, String tipo, String valor) {
      super();
      this.nombre = nombre;
      this.tipo = tipo;
      this.valor = valor;
}

/** devuelve el nombre del atributo
 * @return nombre del atirbuto */
public String getNombre() {return nombre;}

/** establece el nombre del atributo
 * @param nombre : nombre del atributo */
public void setNombre(String nombre) {this.nombre = nombre;}

/**devuelve el tipo del atributo
 * @return tipo del atributo */
public String getTipo() {return tipo;}

/** establece el tipo del atributo
 * @param tipo : tipo del atributo */
public void setTipo(String tipo) { this.tipo = tipo;}

/** devuelve el valor del atributo
 * @return valor del atributo */
public String getValor() { return valor; }

/** establece el valor del atributo
 * @param valor : valor del atributo */
public void setValor(String valor) {this.valor = valor;}

/** devuelve un array de Strings con los posibles valores
 * que puede tomar el atributo dentro del conjunto de datos
 * @return array de los posibles valores del atributo */
public String[] getPosiblesValores() { return posiblesValores;}

/** establece los posibles valores que puede tomar un atributo
 * @param posiblesValores : valores posibles del atributo */
```

```
public void setPosiblesValores(String posiblesValores) {
posiblesValores = posiblesValores.substring(1, posiblesValores.length() -
1);
String[] array = posiblesValores.split(",");
 for (int i = 0; i < array.length; i++)
            array[i] = array[i].trim();
 this.posiblesValores = array;
}

/** devuelve el valor maximo del atributo
 * @return maximo /
public double getMax() { return max;}

/** establece el valor maximo del atributo
 * @param max : valor maximo del atributo */
public void setMax(double max) { this.max = max;}

/** devuelve la media del valor del atributo
 * @return media del atributo */
public double getMedia() { return media; }

/** establece la media del valor del atributo
 * @param media : media del atribut */
public void setMedia(double media) { this.media = media; }

/** devuelve el valor minimo del atributo
 * @return minimo  */
public double getMin() { return min; }

/** establece el valor minimo del atributo
 * @param min : valor minimo del atributo  */
public void setMin(double min) { this.min = min;}

/** devuelve la desviacion estandar del valor del atributo
 * @return desviacion estandar */
public double getStdDev() { return stdDev;}

/**establece la desviacion estandar del valor del atributo
 * @param stdDev : desviacion estandar del valor del atributo */
public void setStdDev(double stdDev) {this.stdDev = stdDev;}
```

```
 /** devuelve un HashMap que contiene los valores que puede tomar
  * un atributo y el numero de ocurrencias de cada valor en
  * el conjunto de datos
  * @return LinkedHashMap */
 public LinkedHashMap getLabels() {
       return labels;
 }

 /** establece los valores que puede tomar un atributo y el numero
  * de ocurrencias de cada valor en el conjunto de datos
  * @param labels : LinkedHashMap que contiene los posibles valores del
atributo
  * y el numero de ocurrencias de cada valor en el conjunto de datos */
 public void setLabels(LinkedHashMap labels) {this.labels = labels;}

}
```

Paquete:com.machine.learning.bean

Clase:DatasetBean.java

```
package com.machine.learning.bean;
import java.util.ArrayList;

public class DatasetBean {

 /** nombre del conjunto de datos*/
 private String nombreDataSet;
 /** lista con todas las instancias del conjunto de datos*/
 private ArrayList<InstanciaBean> instancias;
 /** lista con todos los atributos del conjunto de datos */
 private ArrayList<AtributoBean> atributos;
 /** numero total de atirbutos del conjunto de datos */
 private int numeroAtributos;
 /** numero total de instancias del conjunto de datos*/
 private int numeroInstancias;

 /** Constructor por defecto*/
 public DatasetBean() {
       super();instancias = new ArrayList<InstanciaBean>();
 }
```

```
/** Constructor
 * @param tamaño : numero de instancias del conjunto de datos*/
public DatasetBean(int tamaño) {
      instancias = new ArrayList<InstanciaBean>(tamaño);
}

/** metodo que añade una instancia al conjunto de datos
 * @param i : instancia a añadir al conjunto de datos */
public void addInstancia(InstanciaBean i) {
      instancias.add(i);
}
/** metodo que devuelve una lista de todas las instancias del
 * conjunto de datos
 * @return lista de instancias del conjunto de datos */
public ArrayList<InstanciaBean> getInstancias() {return instancias;}

/** establece las instancias del conjutno de datos
 * @param instancias : instancias asignadas al conjunto de datos */
public void setInstancias(ArrayList<InstanciaBean> instancias) {
      this.instancias = instancias;
}

/** Devuelve un listado de los atributos de un conjunto de datos
 * @return listado de atributos */
public ArrayList<AtributoBean> getAtributos() {return atributos; }

/** establece los atributos de un conjunto de datos
 * @param atributos : listado de atributos que asignamos al conjunto de
 * datos */
public void setAtributos(ArrayList<AtributoBean> atributos) {
      this.atributos = atributos;
}

/** Devuelve el nombre del conjunto de datos
 * @return nombre del conjunto de datos*/
public String getNombreDataSet() {return nombreDataSet;}

/** establece el nombre del conjunto de datos
 * @param nombreDataSet : nombre conjunto de datos */
```

```
 public void setNombreDataSet(String nombreDataSet) {
       this.nombreDataSet = nombreDataSet;
 }

 /** devuelve el numero total de atributos
  * @return numero de atributos */
 public int getNumeroAtributos() {
       numeroAtributos = atributos.size();
       return numeroAtributos;
 }

 /**devuelve el numero total de instancias
  * @return numero de instancias */
 public int getNumeroInstancias() {
       numeroInstancias = instancias.size();
       return numeroInstancias;
 }

 /** metodo que recorre el listado de atributos del conjunto de datos
cargado y
  * un objeto de la clase AtributoBean que representa un atributo, para ello
  * le pasamos como parametro el nombre del atributo a devolver.
  * @param nombreAttr : nombre del atributo que queremos obtener de
  * la lista de atributos.
  * @return un atributo de la clase AtributoBean */
 public AtributoBean getAttrPorNombre(String nombreAttr) {
       AtributoBean res = null;
       for (int i = 0; i < atributos.size(); i++) {
             AtributoBean attr = (AtributoBean) atributos.get(i);
             if (attr.getNombre().equals(nombreAttr.trim())) {
                   res = attr;
             }
       }
       return res;
 }
}
```

Paquete:com.machine.learning.bean

Clase:InstanciaBean.java

```
package com.machine.learning.bean;
import java.util.Hashtable;

public class InstanciaBean {

/** una instancia es una coleccion de atributos con sus correspondientes
valores,  en valoresAtributos se almacena dicha coleccion */
 private Hashtable<String, String> valoresAtributos;
 /** numero de cluster al que pertence esta instancias o muestra */
 private int numCluster;
 /** numero de clase al que pertence esta instancias o muestra */
 private int numClase;

/**constructor por defecto */
 public InstanciaBean() {
       super();
       this.valoresAtributos = new Hashtable<String, String>();
 }

 /**constructor
  * @param tamaño : numero de atributos */
 public InstanciaBean(int tamaño) {
       super();
       this.valoresAtributos = new Hashtable<String, String>(tamaño);
 }

 /** devuelve  el numero de clase al que pertence esta instancias o muestra
  * @return numero clase */
 public int getNumClase() { return numClase;}

 /** establece el numero de clase al que pertence esta instancias o muestra
  * @param numClase */
 public void setNumClase(int numClase) { this.numClase = numClase;}

 /** devuelve la lista de atributos con sus correspondientes valores de la
  * instancia actual
  * @return Hashtable de atributos y sus correspondientes valores */
```

```
 public Hashtable<String, String> getValoresAtributos() {
  return valoresAtributos;
  }

 /**establece la lista de atributos con sus correspondientes valores de la
  * instancia actual
  * @param valoresAtributos : Hashtable que contiene la lista de atributos
*/
   public void setValoresAtributos(Hashtable<String, String>
valoresAtributos) {
       this.valoresAtributos = valoresAtributos;
  }

 /** añade un atributo y su correspondiente valor a la lista de atributos,
  * @param nombreAttr : nombre del atributo
  * @param valorAttr : valor del atributo
  * @return String */
  public String setValue(String nombreAttr, String valorAttr) {
       return valoresAtributos.put(nombreAttr, valorAttr);
  }

 /** devuele el valor de un determinado atributo que pertence a esta
instancia
  * o muestra
  * @param nombreAttr : nombre de atributo el cual queremos obtener su valor
  * @return valor del atributo */
  public String getValue(String nombreAttr) {
       return valoresAtributos.get(nombreAttr);
 }

 /** devuelve el numero de cluster al que pertence esta instancias o muestra
  * @return numero de cluster */
 public int getNumCluster() { return numCluster;}

 /** establece el numero de cluster al que pertence esta instancias o
muestra
  * @param numCluster : numero de cluster */
 public void setNumCluster(int numCluster) { this.numCluster = numCluster;}

 /** metodo toString que devuelve una cadena de texto que describe
  * la instancia actual
  * @return String descripicion del nodo actual */
```

```
 public String toString() {
       String s;
       s = valoresAtributos.toString() + " numero Cluster:" +
this.numCluster;
       return s;
 }

}
```

Paquete:com.machine.learning.bean

Clase:C45Bean.java

```
package com.machine.learning.bean;
import java.util.ArrayList;

public class C45Bean {

 /** numero de clases */
 private int numClases;
 /** arraylist de clases */
 private ArrayList<ClaseBean> clases;
 /** numero de instancias*/
 private double numInstancias;
 /** conjunto de datos utilizado en el entrenamiento del algoritmo */
 private DatasetBean dataset;
 /** nombre del algoritmo */
 private String algoritmo = "C4.5 Tree";
 /** clase indice o discriminatoria , clase a partir de la cual se hace la
  * separacion de las muestras en clases */
 private String claseIndice;
 /** nodo root que contiene todos los nodos del arbol */
 private NodoArbolBean arbol;
 /** graph description language - descripcion grafica del arbol con el
 * lenguaje DOT */
 private String dot;
 /** numero de elementos que hay en el arbol */
 private int numElementos;

 /** Constructor por defecto*/
 public C45Bean() { super(); }
```

```
/** devuelve el nombre del algoritmo
 * @return nombre del algoritmo */
public String getAlgoritmo() { return algoritmo;}

/** devuelve el numero de clases
 * @return numero de clases */
public int getNumClases() { return numClases;}

/** establece el numero de clases
 * @param numClases : numero de clases */
public void setNumClases(int numClases) {this.numClases = numClases;}

/** devuelve un lista de objetos de la clase ClaseBean, que contienen
 * el resultado del entrenamiento del algoritmo C4.5 con un conjunto
 * de datos dado.
 * @return lista de objetos ClaseBean */
public ArrayList<ClaseBean> getClases() { return clases;}

/** establece las clases de un cojunto de datos utilizado en el
 * entrenamiento del algoritmo C4.5
 * @param clases : clases de un cojunto de datos utilizado en el
 * entrenamiento del algoritmo C4.5 */
public void setClases(ArrayList<ClaseBean> clases) { this.clases = clases;}

/** devuelve el numero de instancias del conjunto de datos a clasificar
 * @return numero de instancias */
public double getNumInstancias() {return numInstancias; }

/** establece el numero de instancias del conjunto de datos a clasificar
 * @param numInstancias : numero de instancias */
public void setNumInstancias(double numInstancias) {
      this.numInstancias = numInstancias;
}

/**devuelve el numero de elementos que tiene el arbol C4.5
 * @return numero de elementos*/
public int getNumElementos() { return numElementos;}

/** establece el numero de elementos del arbol
```

```
  * @param numElementos : numero de elementos del arbol c4.5*/
 public void setNumElementos(int numElementos) {
       this.numElementos = numElementos;
 }

 /** devuelve el nombre del atributo utilizado como clase indice
  * para clasificar un conjunto de datos
  * @return nombre del atributo usado como clase indice*/
 public String getClaseIndice() {return claseIndice;}

 /** establece el nombre del atributo utilizado como clase indice
  * para clasificar un cojunto de datos
  * @param claseIndice : nombre del atributo utilizado como clase indice*/
 public void setClaseIndice(String claseIndice) {this.claseIndice =
claseIndice;}

 /** devuelve un arbol C4.5 en forma de un objeto de la clase NodoArbolBean,
  * dicho objeto contiene todos los elementos del arbol C4.5
  * @return NodoArbolBean que representa un arbol C4.5*/
 public NodoArbolBean getArbol() { return arbol;}

 /** establece un arbol que contiene el resultado de la clasificion de un
  * conjunto de datos dado utilizando el algoritmo C4.5 para clasificarlo.
  * @param arbol : arbol C4.5 que contiene el resultado de la clasificacion
  * de un conjunto de datos dado.*/
 public void setArbol(NodoArbolBean arbol) {this.arbol = arbol;}

 /** devuelve el conjunto de datos utilizado en el entrenamiento del
  * algoritmo C4.5
  * @return un conjunto de datos*/
 public DatasetBean getDataset() { return dataset; }

 /** establece el conjunto de datos a utilizar en el entrenamiento
  * del algoritmo C4.5
  * @param dataset : conjunto de datos */
 public void setDataset(DatasetBean dataset) { this.dataset = dataset;}

 /** devuelve una cadena de texto en formato DOT (graph description
language)
  * que describe un arbol C4.5
```

```
   * @return una cadena de texto que describe el arbol c4.5 en notacion DOT
*/
 public String getDot() {
       return dot;
 }

 /** guarda la informacion que describe un arbol C4.5 utilizando una
notacion DOT
   * (graph description language)
   * @param dot : cadena de texto que describe el arbol C4.5 */
 public void setDot(String dot) {
       this.dot = dot;
 }

}
```

Paquete:com.machine.learning.bean

Clase:EMBean.java

```
package com.machine.learning.bean;
import java.util.ArrayList;

public class EMBean {

 /** numero maximo de iteraciones */
 private int maxIteraciones;
 /** numero de clusteres o centroides */
 private int numClusters;
 /** set o conjunto de datos que se utiliza en el entrenamiento del
algoritmo EM */
 private DatasetBean dataset;
 /** nombre del algoritmo */
 private String algoritmo = "EM";
 /** coeficiente de verosimilitud */
 private double logLikelihood;
 /** numero de iteraciones optimos para que el algoritmo converge */
 private int numIteraciones;
 /** numero de instancias que pertenecen a cada cluster */
 private int[] numInstanciasCluster;
 /** lista de clusteres */
 private ArrayList<ClusterBean> clusters;
```

```
 /** Constructor por defecto */
 public EMBean() { super();}

 /** Constructor
  * @param maxIteraciones numero maximo de iteraciones */
 public EMBean(int maxIteraciones) {
       super();
       this.maxIteraciones = maxIteraciones;
 }

 /** devuelve el numero de instancias que pertenecen a cada cluster
  * @return array de numero de instancias de todos los clusteres*/
 public int[] getNumInstanciasCluster() { return numInstanciasCluster;}

 /** establece el numero de instancias que pertenecen a cada uno de los
clusteres
  * @param numInstanciasCluster : array con el numero de instancias que
pertenecen
  * a cada uno de los clusteres */
 public void setNumInstanciasCluster(int[] numInstanciasCluster) {
       this.numInstanciasCluster = numInstanciasCluster;
 }

 /** devuele el valor del coeficiente de verosimilitud (LogLikelihood)
  * @return valor del coeficiente de verosimilitud */
 public double getLogLikelihood() { return logLikelihood;}

 /** devuelve el numero de iteraciones necesarios para que el algoritmo
converge
  * @return numero de iteraciones */
 public int getNumIteraciones() { return numIteraciones; }

 /** establece el numero de iteraciones necesarios para que el algoritmo
converge
  * @param numIteraciones : numero de iteraciones  */
 public void setNumIteraciones(int numIteraciones) {
       this.numIteraciones = numIteraciones;
 }

 /** establece el valor del coeficiente de verosimilitud (LogLikelihood)
  * @param logLikelihood : coeficiente de verosimilitud */
```

```
public void setLogLikelihood(double logLikelihood) {
      this.logLikelihood = logLikelihood;
}

/** devuelve un arraylist que contiene todos los clusteres resultantes
 * del entrenamiento del algoritmo EM
 * @return lista de clusteres */
public ArrayList<ClusterBean> getClusters() { return clusters;}

/** alamacena una lista de todos los clusteres resultantes del
entrenamiento
 * del algoritmo EM
 * @param clusters : arraylist que contiene clusteres */
public void setClusters(ArrayList<ClusterBean> clusters) {
      this.clusters = clusters;
}

/** devuelve el conjunto de datos utilizado en el entrenamiento del
 * algoritmo EM
 * @return conjunto de datos */
public DatasetBean getDataset() { return dataset;}

/** establece el conjunto de datos utilizado en el entrenamiento del
 * algoritmo EM
 * @param dataset : conjunto de datos utilizado en entrenamiento */
public void setDataset(DatasetBean dataset) { this.dataset = dataset; }

/** devuelve el nombre del algoritmo
 * @return nombre del algoritmo */
public String getAlgoritmo() { return algoritmo; }

/** devuelve el numero de clusteres resultante del entrenamiento del
 * algoritmo EM
 * @return numero de clusteres */
public int getNumClusters() { return numClusters;}

/** establece el numero de clusteres resultante del entrenamiento del
 * algoritmo EM
 * @param numClusters : numero de clusteres */
public void setNumClusters(int numClusters) {
      this.numClusters = numClusters;
```

```
 }

 /** establece el nombre del algoritmo
  * @param algoritmo : nombre del algoritmo */
 public void setAlgoritmo(String algoritmo) { this.algoritmo = algoritmo;}

 /** devuelve el numero maximo de iteraciones
  * @return numero maximo de iteraciones */
 public int getMaxIteraciones() { return maxIteraciones;}

 /** establece el numero maximo de iteraciones
  * @param maxIteraciones : numero maximo de iteraciones  */
 public void setMaxIteraciones(int maxIteraciones) {
       this.maxIteraciones = maxIteraciones;
 }

}
```

Paquete:com.machine.learning.bean

Clase:KMEANSBean.java

```
package com.machine.learning.bean;
import java.util.ArrayList;

public class KMeansBean {

 /** numero de centroides o clusteres, variable K */
 private int numClusters;
 /** arraylist con los centorides */
 private ArrayList<InstanciaBean> centroides;
 /** conjunto de datos utilizado en el entrenamiento del algoritmo KMEANS */
 private DatasetBean dataset;
 /** nombre del algoritmo */
 private String algoritmo = "K-MEANS";
 /** suma de errores cuadraticos dentro del cluster */
 private Double squaredError;
 /** numero de iteraciones optimo para que el algoritmo converge */
 private int numIteraciones;
 /** numero de instancias que pertenecen a cada uno de los clusteres */
 private int[] numInstanciasCluster;
```

```
/** Constructor por defecto */
public KMeansBean() {
      super();
      centroides = new ArrayList<InstanciaBean>();
}

/** Constructor
 * @param numCentroides: numero de centroides variable K inicial */
public KMeansBean(int numCentroides) {
      centroides = new ArrayList<InstanciaBean>(numCentroides);
}

/** devuelve el nombre del algoritmo
 * @return nombre del algoritmo */
public String getAlgoritmo() { return algoritmo;}

/** establece el nombre del algoritmo
 * @param algoritmo : nombre del algoritmo */
public void setAlgoritmo(String algoritmo) {this.algoritmo = algoritmo;}

/** devueleve un array con el numero de instancias de cada uno de los
 * clusteres
 * @return array con el numero de instancias de cada uno de los
 * clusteres */
public int[] getNumInstanciasCluster() { return numInstanciasCluster; }

/** establece el numero de instancias de cada uno de los clusteres
 * @param numInstanciasCluster : array con el numero de instancias
 * que pertenecen a cada uno de los clusteres */
public void setNumInstanciasCluster(int[] numInstanciasCluster) {
      this.numInstanciasCluster = numInstanciasCluster;
}

/**devuelve  numero de iteraciones optimo para que el algoritmo converge
 * @return numero de iteraciones */
public int getNumIteraciones() {return numIteraciones;}

/** establece el numero de iteraciones optimo para que el algoritmo
converge
 * @param numIteraciones : numero de iteraciones */
```

```
public void setNumIteraciones(int numIteraciones) {
      this.numIteraciones = numIteraciones;
}

/** devuelve la suma de errores cuadraticos dentro del cluster
 * @return suma de errores cuadraticos */
public Double getSquaredError() { return squaredError;}

/** establece la suma de errores cuadraticos dentro del cluster
 * @param squaredError : suma de errores cuadraticos */
public void setSquaredError(Double squaredError) {
      this.squaredError = squaredError;
}

/** devuelve un objeto DatasetBean que representa el conjunto de datos
 * utilizado en el entrenamiento del algoritmo KMeans
 * @return DatasetBean conjunto de datos */
public DatasetBean getDataset() { return dataset; }

/** establece el conjunto de datos que se utiliza en el entrenamiento del
 * algoritmo K Means
 * @param dataset : conjunto de datos */
public void setDataset(DatasetBean dataset) { this.dataset = dataset;}

/** añade un centroide a la lista de centriodes,y devuelve true
 * en caso de exito y false en caso contrario.
 * @param i centroide que queremos añadir a la lista de centroides
 * @return boolean true en caso de exito y false en caso contrario */
public boolean addCentroide(InstanciaBean i) { return centroides.add(i); }


/** devuelve un arraylist que contiene los centroides resultantes
 * despues de entrenar el algoritmo KMeans
 * @return lista de centroides */
public ArrayList<InstanciaBean> getCentroides() { return centroides; }

/** establece los centroides
 * @param centroides */
public void setCentroides(ArrayList<InstanciaBean> centroides) {
      this.centroides = centroides;
```

```
 }

 /** devuele el numero de clusteres resultante del entrenamiento del
  * algoritmo KMeans
  * @return numero de clusteres */
 public int getNumClusters() { return numClusters;}

 /** establece el numero de clusteres resultante del entrenamiento del
  * algoritmo KMeans
  * @param numClusters : numero de clusteres */
 public void setNumClusters(int numClusters) {this.numClusters =
numClusters;}

 /** metodo toString que devuele una cadena de texto que contiene
  * informacion del entrenamiento del algoritmo KMeans
  * @return String */
 public String toString() {
       String res = null;
       res = " numero de Clusters : " + numClusters + "\t\n";
       for (int i = 0; i < getNumClusters(); i++) {
             res = res + centroides.get(i).toString();
       }
       return res;
 }

}
```

Paquete:com.machine.learning.bean

Clase:NaiveBayesBean.java

```
package com.machine.learning.bean;
import java.util.ArrayList;

public class NaiveBayesBean {

 /** numero de clases */
 private int numClases;
 /** lista de clases */
 private ArrayList<ClaseBean> clases;
 /** conjunto de datos utilizado en el entrenamiento del algoritmo Naive
Bayes */
```

```
 private DatasetBean dataset;
 /** descripcion del objeto NaiveBayesBean en formato texto */
 private String desc;
 /** nombre del algoritmo */
 private String algoritmo = "Naive Bayes";
 /** numero de iteraciones optimo para que el algoritmo converge */
 private int numIteraciones;
 /** numero de instancias suavizado : es el numero de instancias despues de
  * aplicar la estimacion de laplace como tecnica de suavizado (smoothing
technique)*/
 private double numInstanciasSuavizado;
 /** atributo usado como clase indice o discriminatoria, clase a partir de
la
   * cual se hace la separacion de las muestras en clases */
 private String claseIndice;


 /** constructor por defecto */
 public NaiveBayesBean() { super();}


 /**devuelve el nombre del atributo utilizado como clase indice para
clasificar
   * las instancias del conjunto de datos cargado en la aplicacion.
   * @return clase indice */
 public String getClaseIndice() { return claseIndice; }


 /** establece el nombre del atributo utilizado como clase indice para
clasificar
   * las instancias del conjunto de datos cargado en la aplicacion.
   * @param claseIndice */
 public void setClaseIndice(String claseIndice) { this.claseIndice =
claseIndice;}


 /** devuelve el numero de instancias suavizado
   * @return numero de instancias suavizado */
 public double getNumInstanciasSuavizado() { return numInstanciasSuavizado;}


 /** establece el numero de instancias suavizado
   * @param numInstanciasSuavizado : el numero de instancias suavizado */
 public void setNumInstanciasSuavizado(double numInstanciasSuavizado) {
       this.numInstanciasSuavizado = numInstanciasSuavizado;
 }
```

```
/** devuele el nombre del algoritmo
 * @return nombre del algoritmo */
public String getAlgoritmo() { return algoritmo; }

/** devuelve la descripcion del objeto NaiveBayesBean en formato texto
 * @return descripcion del objeto NaiveBayesBean */
public String getDesc() { return desc; }

/** establece la descripcion del objeto NaiveBayesBean en formato texto
 * @param desc : descripcion del objeto NaiveBayesBean en formato texto */
public void setDesc(String desc){ this.desc = desc; }

/** devuelve el numero de iteraciones optimo para que el algoritmo converge
 * @return numero de iteraciones optimo para que el algoritmo converge */
public int getNumIteraciones(){ return numIteraciones;}

/** establece numero de iteraciones optimo para que el algoritmo converge
 * @param numIteraciones : numero de iteraciones */
public void setNumIteraciones(int numIteraciones){
      this.numIteraciones = numIteraciones;
}

/** devuelve el conjunto de datos utilizado en el entrenamiento del
 * algoritmo Naive Bayes
 * @return conjunto de datos */
public DatasetBean getDataset() { return dataset;}

/**establece el conjunto de datos utilizado en el entrenamiento del
 * algoritmo Naive Bayes
 * @param dataset :  conjunto de datos utilizado en el entrenamiento del
 * algoritmo Naive Bayes */
public void setDataset(DatasetBean dataset) { this.dataset = dataset;}

/** devuelve el numero de clases
 * @return numero de clases */
public int getNumClases() { return numClases; }

/** establece el numero de clases
 * @param numClases :numero de clases */
public void setNumClases(int numClases) { this.numClases = numClases; }
```

```
 /** devuelve un lista que contiene todas las clases resultantes
  * del entrenamiento del algoritmo Naive Bayes
  * @return lista de clases */
 public ArrayList<ClaseBean> getClases() { return clases; }

 /** establece las clases resultantes del entrenamiento del algoritmo
  * Naive Bayes
  * @param clases : clases resultantes del entrenamiento del algoritmo
  * Naive Bayes */
 public void setClases(ArrayList<ClaseBean> clases) {
       this.clases = clases;
 }

}
```

Paquete:com.machine.learning.bean

Clase:NodoArbolBean.java

```
package com.machine.learning.bean;
import java.util.ArrayList;

public class NodoArbolBean {

 /** etiqueta descriptiva del nodo actual */
 private String label;
 /** referencia del nodo actual */
 private String referencia;
 /** referencia del nodo padre del nodo actual */
 private String refPadre;
 /** arista que conecta dos nodos entre si */
 private String arista;
 /** un arraylist que contiene una lista de nodos hijos del actual nodo */
 private ArrayList<NodoArbolBean> hijos;
 /** variable para indicar si el nodo actual es final o no , es decir si
  * tiene nodos hijos */
 private boolean nodoFinal;

 /** Constructor por defecto */
 public NodoArbolBean() { super();}
```

```
/** metodo que devuelve true en caso de que el nodo actual no tenga hijos,
 * y false en caso contrario
 * @return boolean */
public boolean isNodoFinal() {
      if (hijos.size() == 0) nodoFinal = true;
      else nodoFinal = false;
      return nodoFinal;
}

/** establece si el nodo tiene hijos o no
 * @param nodoFinal : boolean true en caso de que tenga hijos */
public void setNodoFinal(boolean nodoFinal) { this.nodoFinal = nodoFinal;}

/** devuelve la arista que conecta el nodo actual con otro nodo
 * @return arista */
public String getArista() { return arista; }

/** establece la arista que conecta el actual nodo con otro nodo
 * @param arista : arista que conecta dos nodos */
public void setArista(String arista) { this.arista = arista;}

/** devuelve la referencia del nodo padre del actual nodo
 * @return referencia del nodo padre */
public String getRefPadre() { return refPadre; }

/** establece la referencia del nodo padre del actual nodo
 * @param refPadre : referencia del nodo padre */
public void setRefPadre(String refPadre) { this.refPadre = refPadre;}

/** devuele un arraylist que contiene los nodos hijos del actual
 * nodo
 * @return nodos hijos del actual nodo  */
public ArrayList<NodoArbolBean> getHijos() { return hijos; }

/** establece los nodos hijos del actual nodo
 * @param hijos : arraylist de nodos */
public void setHijos(ArrayList<NodoArbolBean> hijos) { this.hijos = hijos;}

/** devuelve la referencia del nodo actual
```

```
   * @return referenecia del nodo actual */
  public String getReferencia() { return referencia;}

  /** establece la referencia del nodo actual
   * @param referencia : referencia del nodo actual */
  public void setReferencia(String referencia) { this.referencia =
referencia;}

  /** devuelve la etiqueta descriptiva del nodo actual
   * @return etiqueta descriptiva del nodo actual */
  public String getLabel() { return label;}

  /** establece la etiqueta descriptiva del nodo actual
   * @param label : etiqueta descriptiva del nodo actual  */
  public void setLabel(String label) { this.label = label; }

  /** metodo toString que devuelve una cadena de texto que describe
   * el nodo actual
   * @return string descripicion del nodo actual*/
  public String toString() {
        return "Padre:" + this.refPadre + ",Nodo:" + this.referencia
                    + ", label:" + this.label;

  }

}
```

Paquete:com.machine.learning.bean

Clase:ClaseBean.java

```
package com.machine.learning.bean;
import java.util.ArrayList;

public class ClaseBean {

  /** indice de la clase */
  private int indice;
  /** probabilidad de que una determinada instancia del conjunto de
   *  datos pertenezca a la clase actual */
  private double porcentajeProbabilidad;
```

```
 /** lista que contiene los atributos del conjunto de datos a clasificar */
 private ArrayList<AtributoBean> atributos;
 /** numero de instancias que pertenecen a este clase */
 private double numeroInstancias;
 /** numero de instancias suavizado :es el numero de instancias despues de
  *  aplicar la estimacion de laplace como tecnica de suavizado (smoothing
technique) */
 private double numInstanciasSuavizado;
 /** etiqueta descriptiva de la clase */
 private String label;

 /** Constructor por defecto */
 public ClaseBean() { super(); }

 /** Constructor
  * @param probabilidad : probablidad de que una deterimnada instancia del
  * conjunto de datos pertenezca a la clase actual
  * @param atributos : lista de atributos del conjunto de datos a clasificar
*/
 public ClaseBean(int probabilidad, ArrayList<AtributoBean> atributos) {
       super();
       this.porcentajeProbabilidad = probabilidad;
       this.atributos = atributos;
 }

 /** devuelve la etiqueta descriptiva de la clase actual
  * @return etiqueta descriptiva de la clase */
 public String getLabel() { return label; }

 /** establece una etiqueta descriptiva para la clase actual
  * @param label : etiqueta descriptiva de la clase */
 public void setLabel(String label) { this.label = label; }

 /** devuelve el numero de instancias suavizado
  * @return numero de instancias suavizado */
 public double getNumInstanciasSuavizado() {return numInstanciasSuavizado;}

 /** establece el numero de instancias suavizado
  * @param numInstanciasSuavizado : numero de instancias suavizado */
 public void setNumInstanciasSuavizado(double numInstanciasSuavizado) {
       this.numInstanciasSuavizado = numInstanciasSuavizado;
```

```
}

/** devuelve el numero de instancias que pertenecen a este clase
 * @return numero de instancias */
public double getNumeroInstancias() { return numeroInstancias; }

/** establece el numero de instancias que pertenecen a este clase
 * @param numeroInstancias : numero de instancias  */
public void setNumeroInstancias(double numeroInstancias) {
      this.numeroInstancias = numeroInstancias;
}

/** devuelve el indice de la clase
 * @return indice de la clase */
public int getIndice() {return indice;}

/** establece el indice de la clase
 * @param indice : indice de la clase */
public void setIndice(int indice) { this.indice = indice;}

/** devuelve la probabilidad de que una instancia pertenezca a esta clase.
 * @return probabilidad de que una instancia pertenezca a esta clase. */
public double getPorcentajeProbabilidad() {
      this.porcentajeProbabilidad = (this.numeroInstancias /
this.numInstanciasSuavizado);
      return this.porcentajeProbabilidad;
}

/**establece la porbabilidad de que una instancia del conjunto de datos
 * pertenezca a esta clase.
 * @param porcentajeProbabilidad : probabilidad de que una instancia
pertenezca a esta clase */
public void setPorcentajeProbabilidad(double porcentajeProbabilidad) {
      this.porcentajeProbabilidad = porcentajeProbabilidad;
}

/** devuelve un arraylist que contiene los atributos del conjunto de datos
 * que queremos clasificar
 * @return ArrayList lista de atributos */
public ArrayList<AtributoBean> getAtributos() {return atributos;}
```

```
 /** establece los atributos del conjunto de datos que queremos clasificar
  * @param atributos : lista de atributos */
 public void setAtributos(ArrayList<AtributoBean> atributos) {
       this.atributos = atributos;
   }
 }
```

Paquete:com.machine.learning.bean

Clase:ClusterBean.java

```
package com.machine.learning.bean;
import java.util.ArrayList;

public class ClusterBean {

 /** indice del cluster */
 private int indice;
 /** probabilidad de que una instancia pertenezca al cluster actual */
 private double probablidad;
 /** lista que contiene los atributos del conjunto de datos utilizado
  * en el entrenamiento de un algoritmo */
 private ArrayList<AtributoBean> atributos;
 /** numero de instancias que pertenecen a este cluster */
 private double numeroInstancias;

 /** Constructor por defecto */
 public ClusterBean() { super();}

/** Constructor
* @param probabilidad : probabilidad de que una instancia pertenezca al
cluster
* @param atributos : lista de atributos del conjunto de datos utilizado en
* el entrenamiento de un algoritmo */
 public ClusterBean(int probablidad, ArrayList<AtributoBean> atributos) {
       super();
       this.probablidad = probablidad;
       this.atributos = atributos;
 }

 /** devuelve el numero de instancias que pertenecen al cluster actual
```

```
 * @return numero de instancias que pertenecen al cluster actual */
public double getNumeroInstancias() { return numeroInstancias; }

/** establece el numero de instancias que pertenecen al cluster actual
 * @param numeroInstancias :  numero de instancias */
public void setNumeroInstancias(double numeroInstancias) {
      this.numeroInstancias = numeroInstancias;
}

/** devuelve el indice del cluster
 * @return  el indice del cluster */
public int getIndice() { return indice; }

/** establece el indice del cluster
 * @param indice : el indice del cluster */
public void setIndice(int indice) { this.indice = indice;}

/** devuelve la probabilidad de que una instancia pertenezca al cluster
actual
 * @return la probabilidad de que una instancia pertenezca al cluster
actual */
public double getProbablidad() { return probablidad; }

/** establece la probabilidad de que una instancia pertenezca al cluster
actual
 * @param probablidad :probabilidad de que una instancia pertenezca al
cluster
 * actual */
 public void setProbablidad(double probablidad) {
      this.probablidad = probablidad;
 }

/** devuelve un arraylist que contiene los atributos del conjunto de datos
 * que utilizamos en el agupamiento (clustering)
 * @return ArrayList de atributos */
public ArrayList<AtributoBean> getAtributos() { return atributos; }


/** establece los atributos del conjunto de datos que utilizamos en el
 * agrupamiento (clustering)
 * @param atributos : atributos del conjunto de datos utilizado en el
 * agrupamiento (clustering) */
```

```
 public void setAtributos(ArrayList<AtributoBean> atributos) {
       this.atributos = atributos;
 }
}
```

Paquete:com.machine.learning.bean

Clase:EvaluacionClaseBean.java

```
package com.machine.learning.bean;
public class EvaluacionClaseBean {

 /** nombre del algoritmo evaluado */
 private String algoritmo;
 /** coeficiente kappa : es un coeficiente para saber el grado de acuerdo
que hay
  * entre las clases verdaderas de las instancias y la salida del
classificador */
 private double kappa;
 /** precision del algoritmo para clasificar un conjunto de datos */
 private double precision;
 /** numero de instancias clasificadas correctamente */
 private double instanciasBienClasificadas;
 /** numero de instancias clasificadas incorrectamente */
 private double instanciasMalClasificadas;
 /** porcentaje de instancias clasificadas correctamente */
 private double porcentajeBienClasificadas;
 /** porcentaje de instancias clasificadas incorrectamente */
 private double porcentajeMalClasificadas;
 /** numero de instancias del conjunto de datos */
 private int numeroInstancias;
 /** numero de atributos del conjunto de datos */
 private int numeroAtributos;
 /** array de dos dimensiones que contiene la matriz de confusion */
 private double[][] matriz;
 /** etiquetas descriptivas de los posibles valores del atributo usado
  * como clase indice */
 private String[] labelsClaseIndice;
 /** nombre del atributo utilizado como clase indice para clasificar
  *  las instancias del conjunto de datos*/
 private String nombreClaseIndice;
```

```
 /** Constructor por defecto */
 public EvaluacionClaseBean() { super();}

 /** devuelve el nombre del atributo usado como clase indice
  * @return nombre clase indice */
 public String getNombreClaseIndice() { return nombreClaseIndice; }

 /** establece el nombre del atributo usado como clase indice
  * @param nombreClaseIndice : nombre de la clase indice */
 public void setNombreClaseIndice(String nombreClaseIndice) {
       this.nombreClaseIndice = nombreClaseIndice;
 }

 /** devuelve un array que contiene etiquetas descriptivas de los posibles
  * valores del atributo usado como clase indice
  * @return array de etiquetas descriptivas */
 public String[] getLabelsClaseIndice() { return labelsClaseIndice; }

 /** establece las etiquetas descriptivas de los posibles valores del
atributo
  *  usado como clase indice
  * @param labelsClaseIndice : array de etiquetas descriptivas de los
posibles
  * valores del atributo usado como clase indice */
 public void setLabelsClaseIndice(String[] labelsClaseIndice) {
       this.labelsClaseIndice = labelsClaseIndice;
 }

 /** devuelve un array de dos dimensiones que representa la matriz
  * de confusion
  * @return matriz de confusion */
 public double[][] getMatriz() { return matriz;}

 /** establece la matriz de confusion
  * @param matriz : matriz de confusion */
 public void setMatriz(double[][] matriz) { this.matriz = matriz;}


 /** devuelve el nombre del algoritmo
  * @return nombre del algoritmo */
 public String getAlgoritmo() { return algoritmo;}
```

```
/** establece el nombre del algoritmo
 * @param algoritmo : nombre del algoritmo */
public void setAlgoritmo(String algoritmo) { this.algoritmo = algoritmo;}

/** devuelve el coeficiente Kappa
 * @return coeficiente Kappa */
public double getKappa() { return kappa;}

/** establece el coeficiente kappa
 * @param kappa : coeficiente kappa */
public void setKappa(double kappa) { this.kappa = kappa;}

/** devuelve la precision del algoritmo para clasificar un conjunto de
 * datos
 * @return precision del algoritmo */
public double getPrecision() { return precision; }

/** establece la precision del algoritmo para clasificar un conjunto de
 * datos
 * @param precision : precision del algoritmo */
public void setPrecision(double precision) {this.precision = precision;}

/** devuelve el numero de instancias clasificadas correctamente
 * @return numero de instancias bien clasificadas*/
public double getInstanciasBienClasificadas() {
      return instanciasBienClasificadas;
}

/** establece el numero de instancias clasificadas correctamente
 * @param instanciasBienClasificadas : numero de instancias clasificadas
 * correctamente */
public void setInstanciasBienClasificadas(double
instanciasBienClasificadas) {
      this.instanciasBienClasificadas = instanciasBienClasificadas;
}

/** devuelve el numero de instancias clasificadas incorrectamente
 * @return numero de instancias mal clasificadas */
public double getInstanciasMalClasificadas() {
      return instanciasMalClasificadas;
```

```
 }

 /** establece el numero de instancias clasificadas incorrectamente
  * @param instanciasMalClasificadas : numero de instancias mal clasificadas
  */
 public void setInstanciasMalClasificadas(double instanciasMalClasificadas)
{
       this.instanciasMalClasificadas = instanciasMalClasificadas;
 }

 /** devuelve el porcentaje de las instancias clasificadas correctamente
  * @return porcentaje de instancias bien clasificadas */
 public double getPorcentajeBienClasificadas() {
       return porcentajeBienClasificadas;
 }

 /** establece el porcentaje de las instancias clasificadas correctamente
  * @param porcentajeBienClasificadas : porcentaje de instancias
clasificadas correctamente */
 public void setPorcentajeBienClasificadas(double
porcentajeBienClasificadas) {
       this.porcentajeBienClasificadas = porcentajeBienClasificadas;
 }

 /** devuelve el porcentaje de las instancias clasificadas incorrectamente
  * @return porcentaje de instancias mal clasificadas */
 public double getPorcentajeMalClasificadas() {
       return porcentajeMalClasificadas;
 }

 /** establece el porcentaje de las instancias clasificadas incorrectamente
  * @param porcentajeMalClasificadas : porcentaje de instancias clasificadas
incorrectamente */
 public void setPorcentajeMalClasificadas(double porcentajeMalClasificadas)
{
       this.porcentajeMalClasificadas = porcentajeMalClasificadas;
 }

 /** devuelve el numero de instancias del conjunto de datos
  * @return numero de instancias del conjunto de datos */
 public int getNumeroInstancias() { return numeroInstancias;}
```

```
 /** establece el numero de instancias del conjunto de datos
  * @param numeroInstancias : numero de instancias del conjunto de datos */
 public void setNumeroInstancias(int numeroInstancias) {
       this.numeroInstancias = numeroInstancias;
 }

 /** devuele el numero de atributos del conjunto de datos
  * @return numero de atributos */
 public int getNumeroAtributos() { return numeroAtributos; }

 /** establece el numero de atributos del conjunto de datos
  * @param numeroAtributos : numero de atributos del conjunto de datos
  */
 public void setNumeroAtributos(int numeroAtributos) {
       this.numeroAtributos = numeroAtributos;
 }

}
```

Paquete:com.machine.learning.bean

Clase:EvaluacionClusterBean.java

```
package com.machine.learning.bean;
public class EvaluacionClusterBean {

 /** numero de iteraciones optimo para que el algoritmo converge */
 private int numeroIteraciones;
 /** suma de errores cuadraticos dentro del cluster */
 private double sumaErrores;
 /** coeficiente de verosimilitud */
 private double logLikelihood;
 /** numero de clusteres */
 private int numeroClusters;
 /** numero de instancias que pertenecen a cada cluster */
 private int[] numInstanciasCluster;
 /** porcentaje de las instancias que pertenecen a cada uno de los clusteres
*/
 private double[] porcentajeInstanciasCluster;
 /** numero de instancias que contiene el dataset o conjunto de datos */
 private int numInstancias;
```

```
/** Constructor por defecto */
public EvaluacionClusterBean() { super();}

/** devuelve el porcentaje de las instancias que pertenecen a cada uno de
 * los clusteres
 * @return porcentaje de las instancias que pertenecen a cada uno de
 * los clusteres */
public double[] getPorcentajeInstanciasCluster() {
      return porcentajeInstanciasCluster;
}

/** establece el porcentaje de las instancias que pertenecen a cada uno de
 * los clusteres
 * @param porcentajeInstanciasCluster : porcentaje de las instancias que
 * pertenecen a cada uno de los clusteres */
public void setPorcentajeInstanciasCluster(
            double[] porcentajeInstanciasCluster) {
      this.porcentajeInstanciasCluster = porcentajeInstanciasCluster;
}

/** devuelve el coeficiente de verosimilitud
 * @return coeficiente de verosimilitud */
public double getLogLikelihood() { return logLikelihood; }

/** establece el coeficiente de verosimilitud
 * @param logLikelihood : coeficiente de verosimilitud */
public void setLogLikelihood(double logLikelihood) {
      this.logLikelihood = logLikelihood;
}

/** devuelve el numero de iteraciones optimo para que el
 * algoritmo converge
 * @return numero de iteraciones  */
public int getNumeroIteraciones() { return numeroIteraciones;}

/** establece el numero de iteraciones optimo para que el
 * algoritmo converge
 * @param numeroIteraciones : numero de iteraciones */
public void setNumeroIteraciones(int numeroIteraciones) {
```

```
		this.numeroIteraciones = numeroIteraciones;
	}

	/** devuelve la suma de errores cuadraticos dentro del cluster
	 * @return suma de errores cuadraticos dentro del cluster */
	public double getSumaErrores() { return sumaErrores; }

	/** establece la suma de errores cuadraticos dentro del cluster
	 * @param sumaErrores : suma de errores cuadraticos dentro del cluster */
	public void setSumaErrores(double sumaErrores) { this.sumaErrores =
sumaErrores;}

	/** devuelve el numero de clusteres
	 * @return numero de clusteres  */
	public int getNumeroClusters() { return numeroClusters; }

	/** establece el numero de clusteres
	 * @param numeroClusters : numero de clusteres  */
	public void setNumeroClusters(int numeroClusters) { this.numeroClusters =
numeroClusters;}

	/** devuelve el numero de instancias que pertenecen a cada cluster
	 * @return numero de instancias que pertenecen a cada cluster */
	public int[] getNumInstanciasCluster() { return numInstanciasCluster;}

	/** establece el numero de instancias que pertenecen a cada cluster
	 * @param numInstanciasCluster : numero de instancias que pertenecen a cada
cluster */
	public void setNumInstanciasCluster(int[] numInstanciasCluster) {
		this.numInstanciasCluster = numInstanciasCluster;
	}

	/** devuelve el numero de instancias del conjunto de datos
	 * @return el numero de instancias */
	public int getNumInstancias() { return numInstancias;}

	/** establece el numero de instancias del conjunto de datos
	 * @param numInstancias : el numero de instancias */
	public void setNumInstancias(int numInstancias) {
		this.numInstancias = numInstancias;
	}}
```

Paquete:com.machine.learning.controller

Clase:ControladorSeleccion.java

```
package com.machine.learning.controller;
import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;
import
org.springframework.web.servlet.mvc.multiaction.MultiActionController;
import com.machine.learning.bean.DatasetBean;
import com.machine.learning.bo.ProcesadorDatos;


@Controller
@RequestMapping("/seleccion")
public class ControladorSeleccion extends MultiActionController{

 /**
  * metodo que gestiona las solicitudes de clientes via http tipo GET
  * a la siguiente url '/seleccion'
  * @return String
  */
 @RequestMapping(method = RequestMethod.GET)
    public String verFormulario () {
        return "seleccion";
    }


    /**
     * metodo que gestiona las solicitudes de clientes via http tipo POST
     * a la siguiente url '/seleccion'
     * @param request  : solicitud http del cliente
     * @param response : respuesta http del servidor
     * @return ModelAndView : objeto que contiene la informacion devuelta
```

```
     *  por el servidor.
     */
    @RequestMapping(method = RequestMethod.POST)
    public ModelAndView procesarFichero(final HttpServletRequest request,
                                    final HttpServletResponse response)  {

    ModelAndView model = new ModelAndView("seleccion");
    String mensajeError="";
    DatasetBean  set = new DatasetBean();
    ProcesadorDatos pd = new ProcesadorDatos();

    try {
       InputStream iss = request.getInputStream();
       BufferedReader br = new BufferedReader(new InputStreamReader(iss));

       //lee el fichero y lo transforma en un objeto de la clase DataSetBean
       set = pd.leerfichero(br);
      // almacenamos la siguientes variables en la session
      request.getSession().setAttribute("dataSet", set);
      request.getSession().setAttribute("listaAtributos",
set.getAtributos());
       model.addObject("dataSet",set);

   } catch (Exception e) {
     e.printStackTrace();
    mensajeError = "Se ha producido el siguiente error y no se ha podido
subir "                 +"el fichero con exito :<BR> "+e.getMessage();

             model.addObject("mensajeErrorSeleccion",mensajeError);

             request.getSession().setAttribute("dataSet", null);
             request.getSession().setAttribute("listaAtributos", null);
        }

        return model;
    }
}
```

Paquete:com.machine.learning.controller

Clase:ControladorProcesado.java

```
package com.machine.learning.controller;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;
import
org.springframework.web.servlet.mvc.multiaction.MultiActionController;
import com.machine.learning.bean.AtributoBean;
import com.machine.learning.bean.DatasetBean;
import com.machine.learning.bo.ProcesadorDatos;

@Controller
@RequestMapping("/formularioProcesado")
public class ControladorProcesado extends MultiActionController{

 /**
  * metodo que gestiona las solicitudes de clientes via http tipo GET
  * a la siguiente url '/formularioProcesado'
  * @return String
  */
 @RequestMapping(method = RequestMethod.GET)
    public String verFormulario () {
        return "preprocesado";
    }


 /**
     * metodo que gestiona las solicitudes de clientes via http tipo POST
     * a la siguiente url '/formularioProcesado'
     * @param request  : solicitud http del cliente
     * @param response : respuesta http del servidor
     * @return ModelAndView : objeto que contiene la informacion devuelta
     *  por el servidor.
     */
    @RequestMapping(method = RequestMethod.POST)
```

```
    public ModelAndView procesarFichero(final HttpServletRequest request,
                                        final HttpServletResponse response){

       ModelAndView model = new ModelAndView("preprocesado");
       String mensajeError="";
       DatasetBean set = null;
       ProcesadorDatos pd = new ProcesadorDatos();
        AtributoBean attr_aux = new AtributoBean();


    try {
        set = (DatasetBean) request.getSession().getAttribute("dataSet");
        String selAtributos = (String)request.getParameter("selAtributos");


      // comprobamos que existe un conjunto de datos cargado en la
aplicacion
            if(set!=null){
                  AtributoBean attr =
set.getAttrPorNombre(selAtributos.trim());
        attr_aux = pd.setPropertiesToAttr(attr,set.getInstancias());

// en caso contrario mostramos un error
 }else{
   mensajeError = "Se ha producido el siguiente error al procesar el
atributo :"                    +"<BR><BR> No hay ningun conjunto de datos
cargado en la aplicacion.";
}
 } catch (Exception e) {
     e.printStackTrace();
     mensajeError = "Se ha producido el siguiente error al procesar el
atributo:"
                  +"<BR> "+e.getMessage();
    }


        model.addObject("selAttr",attr_aux);
        model.addObject("dataSet2",set);
        model.addObject("mensajeErrorProcesado",mensajeError);
        return model;
    }
}
```

Paquete:com.machine.learning.controller
Clase:ControladorPatrones.java

```
package com.machine.learning.controller;
import java.util.ArrayList;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;
import com.machine.learning.bean.AtributoBean;
import com.machine.learning.bean.DatasetBean;
import com.machine.learning.bean.EMBean;
import com.machine.learning.bean.C45Bean;
import com.machine.learning.bean.KMeansBean;
import com.machine.learning.bean.NaiveBayesBean;
import com.machine.learning.bo.EntrenarAlgoritmos;

@Controller
@RequestMapping("/formularioPatrones")
public class ControladorPatrones {

 /**
  * metodo que gestiona las solicitudes de clientes via http tipo GET
  * a la siguiente url '/formularioPatrones'
  * @return String
  */
    @RequestMapping(method = RequestMethod.GET)
    public String verFormulario () {
        return "patrones";
    }


    /**
     * metodo que gestiona las solicitudes de clientes via http tipo POST
     * a la siguiente url '/formularioPatrones'
     * @param request  : solicitud http del cliente
     * @param response : respuesta http del servidor
```

```
     * @return ModelAndView : objeto que contiene la informacion devuelta
     *  por el servidor.
     */
    @RequestMapping(method = RequestMethod.POST)
    public ModelAndView procesarFormulario(final HttpServletRequest request,
    final HttpServletResponse response) throws Exception {

ModelAndView model = new ModelAndView("patrones");
HttpSession sesion = request.getSession();
ArrayList<AtributoBean>  listaAttr =
(ArrayList<AtributoBean>) sesion.getAttribute("listaAtributos");
DatasetBean set = (DatasetBean)
request.getSession().getAttribute("dataSet");

String mensajeError = "";
String tecnica = (String)request.getParameter("mineria");
String algoritmo = "";

 if(tecnica.equals("CLA")){
       algoritmo = (String)request.getParameter("algoritmoCLA");
  }else if (tecnica.equals("CLU")){
        algoritmo = (String)request.getParameter("algoritmoCLU");
  }

EntrenarAlgoritmos eA = new EntrenarAlgoritmos();
KMeansBean KM =null;
EMBean EM = null;
C45Bean ID3 = null;
NaiveBayesBean NB = null;

try{

// comprobamos que existe un conjunto de datos cargado en la aplicacion
 if(set==null || listaAttr==null || listaAttr.size()<=0){
   mensajeError = "Se ha producido el siguiente error al intentar entrenar "
   + "los datos con el algoritmo seleccionado : <BR><BR> "

   + "No hay ningun conjunto de datos cargado en la aplicacion.";
}else{
// Tecnicas de classificacion
        if(tecnica.equals("CLA")){
```

```
       String claseIndice = (String)request.getParameter("claseIndice");
        int iclaseIndice = Integer.parseInt(claseIndice);
        model.addObject("claseIndice",claseIndice);
        if(algoritmo.equals("TREE")){
             ID3 = eA.J48(set,listaAttr,iclaseIndice);
       }else if(algoritmo.equals("NB")){
             NB = eA.NaiveBayes(set,listaAttr,iclaseIndice);
        }

       // Tecnicas de Clustering - agrupamiento
        }else if(tecnica.equals("CLU")){
             String numClusteres =
(String)request.getParameter("numClusters");
             String maxIters =
(String)request.getParameter("maxIteraciones");
             int inumClusteres = Integer.parseInt(numClusteres);
             int imaxIters = Integer.parseInt(maxIters);

             model.addObject("numClusts",numClusteres);
             model.addObject("maxIteras",maxIters);

             if(algoritmo.equals("KM")){
                   KM = eA.KMeans(set,listaAttr,inumClusteres,imaxIters);

                         }else if(algoritmo.equals("EM")){

                               EM =
eA.EM(set,listaAttr,inumClusteres,imaxIters);
                         }
                   }
             }

        }catch(Exception e){
             e.printStackTrace();
              mensajeError = "Se ha producido el siguiente error al intentar
             entrenar los "
             + "datos con el algoritmo seleccionado :<BR> "+e.getMessage();
        }

        model.addObject("algoritmo",algoritmo);
        model.addObject("tecnica",tecnica);
```

```
        model.addObject("kmeans",KM);
        model.addObject("em",EM);
        model.addObject("id3",ID3);
        model.addObject("nb",NB);
        model.addObject("mensajeErrorPatrones",mensajeError);
        return model;

    }
}
```

Paquete:com.machine.learning.controller

Clase:ControladorEvaluacion.java

```
package com.machine.learning.controller;
import java.util.ArrayList;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;
import com.machine.learning.bean.DatasetBean;
import com.machine.learning.bean.EvaluacionClaseBean;
import com.machine.learning.bean.EvaluacionClusterBean;
import com.machine.learning.bo.EvaluarAlgoritmos;

@Controller
@RequestMapping("/formEval")
public class ControladorEvaluacion {

 /**
  * metodo que gestiona las solicitudes de clientes via http tipo GET
     * a la siguiente url '/formEval'
  * @return String
  */
  @RequestMapping(method = RequestMethod.GET)
  public String verFormulario () {
      return "evaluacion";
  }
```

```
  /**
  * metodo que gestiona las solicitudes de clientes via http tipo POST
     * a la siguiente url '/formEval'
     * @param request  : solicitud http del cliente
     * @param response : respuesta http del servidor
     * @return ModelAndView : objeto que contiene la informacion devuelta
     *  por el servidor.
  */
  @RequestMapping(method = RequestMethod.POST)
  public ModelAndView procesarFormulario(final HttpServletRequest request,
                                                                  final
HttpServletResponse response)
      throws Exception {

      ModelAndView model = new ModelAndView("evaluacion");
      String tecnica = (String)request.getParameter("tecnicaEval");

      String algoritmo = "";

      if(tecnica.equals("CLA")){
              algoritmo = (String)request.getParameter("algoritmoEvalCLA");
      }else if (tecnica.equals("CLU")){
              algoritmo = (String)request.getParameter("algoritmoEvalCLU");
      }

      String mensajeError = "";
      HttpSession sesion = request.getSession();
      DatasetBean set = (DatasetBean) sesion.getAttribute("dataSet");

      EvaluarAlgoritmos eD = new EvaluarAlgoritmos();
      ArrayList<EvaluacionClusterBean> listaEval = null;
      EvaluacionClaseBean evalClase = null;

      try{
  // comprobamos que existe un conjunto de datos cargado en la aplicacion
  if(set==null){
  mensajeError = "Se ha producido el siguiente error al intentar evaluar "
         + "los datos con el algoritmo seleccionado :<BR><BR> "
        + "No hay ningun conjunto de datos cargado en la aplicacion.";
```

```
         }else{
        // Tecnicas de clustering
         if(tecnica.equals("CLU")){
           // algoritmo EM
              if(algoritmo.equals("EM")){

              listaEval = eD.EvaluarAlgoritmoEM(set);
              model.addObject("var_ctrl_eval","EM");

              // algoritmo KMeans
         }else if(algoritmo.equals("KM")){
            listaEval = eD.EvaluarAlgoritmoKMEANS(set);
         model.addObject("var_ctrl_eval","KM");}
                    //    Tecnicas de clasificacion
                    }else  if(tecnica.equals("CLA")){

 String claseIndiceEval = (String)request.getParameter("claseIndiceEval");
 int iclaseIndice = Integer.parseInt(claseIndiceEval);
iclaseIndice = iclaseIndice-1;
 model.addObject("claseIndiceEval",claseIndiceEval);

 // algoritmo Naive Bayes
 if(algoritmo.equals("NB")){
 evalClase = eD.EvaluarAlgoritmoNaiveBayes(set,iclaseIndice);
 model.addObject("var_ctrl_eval","NB");

 // algoritmo de arbol en este caso C4.5
 }else if(algoritmo.equals("TREE")){
 evalClase = eD.EvaluarAlgoritmoTree(set,iclaseIndice);
 model.addObject("var_ctrl_eval","TREE");
                         }
                    }
              }
          }catch(Exception e){
              e.printStackTrace();
              mensajeError = "Se ha producido el siguiente error al intentar"
              +" evaluar los datos con el algoritmo seleccionado :"

              +"<BR> "+e.getMessage();
          }
          model.addObject("listaEval",listaEval);
```

```
        model.addObject("evalClase",evalClase);
        model.addObject("algoritmoEval",algoritmo);
        model.addObject("tecnicaEval",tecnica);
        model.addObject("mensajeErrorEvaluacion",mensajeError);
        return model;
    }
}
```

Paquete:com.machine.learning.bo

Clase:ProcesadorDatos.java

```
package com.machine.learning.bo;
import com.machine.learning.bean.AtributoBean;
import com.machine.learning.bean.DatasetBean;
import com.machine.learning.bean.InstanciaBean;
import java.io.BufferedReader;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Hashtable;
import java.util.LinkedHashMap;

public class ProcesadorDatos {

 /**
  * constructor de la clase
  */
 public ProcesadorDatos() {
       super();
 }

 /**
  * Metodo que se encarga de alimentar todos los atributos de un cojunto de
* datos con propiedades extraidas de las instancias del conjunto
* de datos, se encarga de calcular los valores de minimo, maximo, media,
varianza,
* desviacion estandar para cada uno de los atributos.
*
* @param ArrayList<AtributoBean> : lista de atributos cuyos propiedades
queremos
* calcular
```

```
* @param ArrayList<InstanciaBean>: una lista de instancias de donde se
extrea la
* informacion para calcular las propiedades de los atributos
* @return ArrayList<AtributoBean> : una lista de objetos de la clase
AtributoBean
* que contiene toda la informacion para cada uno de los atributos y sus
propiedades
*/

 public ArrayList<AtributoBean> setPropertiesToAttrs(
             ArrayList<AtributoBean> attrs, ArrayList<InstanciaBean>
instancias){

       ArrayList<AtributoBean> atributos_aux = new
ArrayList<AtributoBean>();

       for (int i1 = 0; i1 < attrs.size(); i1++) {
             AtributoBean attr = attrs.get(i1);
             AtributoBean attr_aux = setPropertiesToAttr(attr, instancias);
             atributos_aux.add(attr_aux);
       }

       return atributos_aux;

 }

/**
* Metodo que se encarga de alimentar un determinado atributo de un cojunto
de
* datos con propiedades extraidas de las instancias del conjunto de datos,
* se encarga de calcular los valores de minimo, maximo, media, varianza,
* desviacion estandar para dicho atributo.
*
* @param AtributoBean : atributo cuyos propiedades queremos calcular
* @param ArrayList<InstanciaBean> : una lista de instancias de donde se
extrea la
* informacion para calcular las propiedades del atributo
* @return AtributoBean : un objeto de la clase AtributoBean que contiene
toda la
* informacion de un atributo y sus propiedades
*/

 public AtributoBean setPropertiesToAttr(AtributoBean atributo,
```

```
            ArrayList<InstanciaBean> instancias) {

      String nombre_attr = atributo.getNombre();
      String tipo_attr = atributo.getTipo();
      ArrayList<Double> lista_aux_numeric = new ArrayList<Double>();
      ArrayList<String> lista_aux_nominal = new ArrayList<String>();

      // para cada instancia
      for (int i2 = 0; i2 < instancias.size(); i2++) {

            InstanciaBean instancia = instancias.get(i2);
            Hashtable ht = instancia.getValoresAtributos();
            String valor = ht.get(nombre_attr).toString();

            if (!valor.equals("?")) {
                  if (tipo_attr.equals("NUMERIC"))
                        lista_aux_numeric.add(Double.parseDouble(valor));
                  else
                        lista_aux_nominal.add(valor);
            }
      }

      // ordenamos la lista
      Collections.sort(lista_aux_numeric);

      if (tipo_attr.equals("NUMERIC")) {

            // calcular la media
            double media = getMean(lista_aux_numeric,
lista_aux_numeric.size());
            atributo.setMedia(media);

            // calcular la desviacion estandar
            double stdDev = getStdDev(lista_aux_numeric, media,
                        lista_aux_numeric.size());
            atributo.setStdDev(stdDev);

            // calcular el minimo
            Double min = Collections.min(lista_aux_numeric);
            atributo.setMin(min);
```

```
            // calcular el maximo
            Double max = Collections.max(lista_aux_numeric);
            atributo.setMax(max);

 LinkedHashMap<Double, String> labels_aux1 = new LinkedHashMap<Double,
String>();

            for (int i3 = 0; i3 < lista_aux_numeric.size(); i3++) {
                  double key = lista_aux_numeric.get(i3);
                  String value = String.valueOf(Collections.frequency(
                              lista_aux_numeric, key));
                  labels_aux1.put(key, value);
            }

            atributo.setLabels(labels_aux1);

      } else if (tipo_attr.equals("NOMINAL")) {

            // elegimos un linkedHashMap porque es un mapa de datos
ordenado, el
            // hashtable no es ordenado
            LinkedHashMap<String, String> labels_aux = new
LinkedHashMap<String, String>();

            for (int i3 = 0; i3 < atributo.getPosiblesValores().length;
i3++) {
                  String key = atributo.getPosiblesValores()[i3];
                  String value = String.valueOf(Collections.frequency(
                              lista_aux_nominal, key));
                  labels_aux.put(key, value);
            }
            atributo.setLabels(labels_aux);
      }
      return atributo;
 }

 /**
  * metodo para obtener la media de una serie de numeros pasados por
  * parametro a traves de un array
  *
  * @param data : array de numeros de tipo double
  * @param size : tamaño del array
```

```
 * @return la media del conjunto
 */

public double getMean(ArrayList<Double> data, double size) {
      double sum = 0.0;
      for (int i = 0; i < data.size(); i++) {
            double a = data.get(i);
            sum += a;
      }
      return sum / size;
}

/**
 * Metodo para obtener la varianza de una serie de numeros pasados por
 * parametro a traves de un array
 * @param data : array de numeros de tipo double
 * @param mean : la media
 * @param size : tamaño del array
 * @return la varianza del conjunto
 */

public double getVariance(ArrayList<Double> data, double mean, double
size){
      double temp = 0;
      for (int i = 0; i < data.size(); i++) {
            double a = data.get(i);
            temp += (mean - a) * (mean - a);
      }
      return temp / size;
}

/**
 * Metodo para obtener la desviacion estandar de una serie de numeros
 * pasados por parametro a traves de un array
 *
 * @param data : array de numeros de tipo double
 * @param mean : la media
 * @param size : tamaño del array
 * @return la desviacion estandar del conjunto
 */
```

```
public double getStdDev(ArrayList<Double> data, double mean, double size){
	return Math.sqrt(getVariance(data, mean, size));
}

/**
 * Lee el contenido de un BufferedReader que pasamos como parametro de
 * entrada, y devuelve un conjunto de datos con sus correspondientes
 * instancias y atributos guardados en un objeto de la clase DatasetBean.
 *
 * @param BufferedReader : objeto que contiene la informacion contenida
 * en un fichero de texto
 * @return DatasetBean : un conjunto de datos con sus correspondientes
 * instancias y atributos
 * @throws Exception lanza una excepcion generica
 */

public DatasetBean leerfichero(BufferedReader br) throws Exception {

	DatasetBean set = new DatasetBean();

	String cadena = "";
	ArrayList<String> lista = new ArrayList<String>();
	ArrayList<AtributoBean> atributos = new ArrayList<AtributoBean>();
	ArrayList<InstanciaBean> instancias = new ArrayList<InstanciaBean>();

	// leemos el fichero quitamos las lineas vacias y los comentarios y
lo
	// almacenamos en un ArrayList
	while ((cadena = br.readLine()) != null) {

		cadena = cadena.trim();
		if (!cadena.startsWith("%") && cadena.length() > 0) {
			lista.add(cadena);
		}
	}

	// varibale para indicar donde comienzan las instancias
	int posicionDatos = lista.indexOf("@data".trim());
	if (posicionDatos == -1)
		posicionDatos = lista.indexOf("@DATA".trim());
```

```
	// no contiene la variable @data , no podemos saber la posicion de
los
	// datos
	if (posicionDatos == -1) {
		throw new Exception(
		  "el fichero seleccionado no contiene la variable @DATA "
		+ " que indica la posicion de los datos dentro del fichero.");
	}

	// variable para indicar la posicion del nombre del fichero
	boolean relation = false;
	for (int i = 0; i < posicionDatos; i++) {
		String relacion = (String) lista.get(i);
		if (relacion.startsWith("@relation")
				|| relacion.startsWith("@RELATION")) {
			String array_relacion[] = relacion.split("\\s+");
			set.setNombreDataSet(array_relacion[1]);
			relation = true;
		}
	}

	// no se ha encontrado la variable @relation
	if (!relation) {
		throw new Exception(
				"el fichero seleccionado no contiene la variable
@RELATION"
					+ " que indica el nombre del conjunto de
datos.");
	}

	// bucle atributos - bucle para construir cada atributo y
introducirlos en un
	// arraylist
	int numeroAtributos = 0;
	for (int i = 0; i < posicionDatos; i++) {
		String as = (String) lista.get(i);
		if (as.startsWith("@attribute") || as.startsWith("@ATTRIBUTE"))
{
			numeroAtributos++;
			String array_attr[] = as.split("\\s+");
			AtributoBean attr = new AtributoBean();
			attr.setNombre(array_attr[1].trim());
```

```
                String tipoAtributo = array_attr[2].toLowerCase();

            if (tipoAtributo.equals("numeric")||
tipoAtributo.equals("real"))
                  attr.setTipo("NUMERIC");
                else
                  attr.setTipo("NOMINAL");

                // posibles valores en caso de que el tipo de attributo
sea
                // string
                if (attr.getTipo().equals("NOMINAL")) {
                  String y = as.substring(as.indexOf("{")).trim();
                  attr.setPosiblesValores(y);
                }

                atributos.add(attr);
            }
      }

      // el fichero seleccionado no contiene atributos
      if (numeroAtributos == 0) {
            throw new Exception(
                  "el fichero seleccionado no contiene variables de tipo
@ATTRIBUTE"
             + " que indican la posicion de los atributos dentro del
fichero.");
      }

      // bucle instancias - convertimos cada linea que viene despues de la
      // etiqueta @data en una instancia nueva
      for (int k = 0; k < lista.size() - posicionDatos - 1; k++) {
      String[] arrayDato = lista.get(posicionDatos + 1 + k).split(",");
            InstanciaBean instancia = new InstanciaBean();
            for (int ia = 0; ia < atributos.size(); ia++) {
                  instancia.setValue(atributos.get(ia).getNombre(),
                              arrayDato[ia].trim());
            }
            instancias.add(instancia);
      }
      // almacenar la lista de atributos en el dataset
      set.setAtributos(atributos);
```

```
        set.setInstancias(instancias);
        return set;
 }
}
```

Paquete:com.machine.learning.bo

Clase:EntrenarAlgoritmos.java

```
package com.machine.learning.bo;
import java.util.ArrayList;
import com.machine.learning.bean.AtributoBean;
import com.machine.learning.bean.ClaseBean;
import com.machine.learning.bean.DatasetBean;
import com.machine.learning.bean.EMBean;
import com.machine.learning.bean.InstanciaBean;
import com.machine.learning.bean.C45Bean;
import com.machine.learning.bean.KMeansBean;
import com.machine.learning.bean.NaiveBayesBean;
import com.machine.learning.bean.NodoArbolBean;
import com.machine.learning.connectAPI.ClassifierData;
import com.machine.learning.connectAPI.ClusteringData;

public class EntrenarAlgoritmos {

 /**
  * constructor de la clase
  */
 public EntrenarAlgoritmos() {
        super();
 }

/**
* Metodo que ejecuta el algoritmo Naive Bayes y devuelve un objeto de
* clase NaiveBayesBean que contiene el resultado del entrenamiento de dicho
* algoritmo utilizando un conjunto de datos, como parametros le pasamos
* el conjunto de datos a utilizar en el entrenamiento y una lista con los
* atributos de dicho conjunto de datos y el atributo utilizado como clase
* indice.
*
* @param set : conjunto de datos utilizado en el entrenamiento del algoritmo
```

```
*  Naive Bayes
* @param attrs : lista de atributos del conjunto de datos
* @param claseIndice : el atributo utilizado como clase indice para
clasificar
* @return NaiveBayesBean : un objeto que contiene el resultado del
entrenamiento
*               del algoritmo naive bayes
* @throws Exception lanza una excepcion generica
*/

 public NaiveBayesBean NaiveBayes(DatasetBean set,
             ArrayList<AtributoBean> attrs, int claseIndice) throws
Exception {

       NaiveBayesBean nB = new NaiveBayesBean();
       String algoritmo = ClassifierData.ALGORITMO_NAIVE_BAYES;
       claseIndice = claseIndice - 1;
       // constriumos el objeto classificador
       ClassifierData cd = new ClassifierData();
       Object NBClassifier = cd.Classifier(set, claseIndice, algoritmo);
       // nombre clase indice
       String nombreIndice = attrs.get(claseIndice).getNombre();
       nB.setClaseIndice(nombreIndice);
       // class labels
       String[] Classlabels = attrs.get(claseIndice).getPosiblesValores();
       nB.setNumClases(Classlabels.length);
       // numero total de instancias suavizado

 nB.setNumInstanciasSuavizado(cd.getNumInstanciasSuavizado(NBClassifier));
       ArrayList<ClaseBean> lista_clases = cd.getDistribucionClases(
                   NBClassifier, Classlabels, claseIndice);
       nB.setClases(lista_clases);
       // asginar instancias a sus correspondientes clases
       DatasetBean dataset = cd.asignarInstanciasToClases(NBClassifier,
                   algoritmo);
       dataset.setAtributos(attrs);
       nB.setDataset(dataset);
       return nB;
 }


/**
```

```
* Metodo que ejecuta el algrotimo C4.5 y devuelve un objeto de clase
* C45Bean que contiene el resultado del entrenamiento de dicho
* algoritmo utilizando un conjunto de datos, como parametros le pasamos
* el conjunto de datos a utilizar en el entrenamiento y una lista con los
* atributos de dicho conjunto de datos y el atributo utilizado como clase
* indice.
*
* @param set : conjunto de datos utilizado en el entrenamiento del algoritmo
C4.5
* @param attrs : lista de atributos del conjunto de datos
* @param claseIndice : el atributo utilizado como clase indice para
clasificar
* @return C45Bean : objeto que contiene el resultado del entrenamiento
* del algoritmo C4.5
* @throws Exception lanza una excepcion generica
*/
 public C45Bean J48(DatasetBean set, ArrayList<AtributoBean> attrs,
             int claseIndice) throws Exception {

       C45Bean j48 = new C45Bean();
       String algoritmo = ClassifierData.ALGORITMO_TREE_J48;
       claseIndice = claseIndice - 1;
       // constriumos el objeto classificador
       ClassifierData cd = new ClassifierData();
       Object TreeClassifier = cd.Classifier(set, claseIndice, algoritmo);
       // establecemos las clase indice o discriminatoria
       String nombreclaseIndice = attrs.get(claseIndice).getNombre();
       j48.setClaseIndice(nombreclaseIndice);
       // numero instancias
       j48.setNumInstancias(set.getInstancias().size());
       // class labels
       String[] Classlabels = attrs.get(claseIndice).getPosiblesValores();
       j48.setNumClases(Classlabels.length);
       // dot graph
       String graph = cd.getDotGraph(TreeClassifier);
       j48.setDot(graph);
       // arbol de nodos
       NodoArbolBean arbol = cd.getArbol(TreeClassifier);
       j48.setArbol(arbol);
       // distribucion de instancias en sus correspondientes clases
       ArrayList<ClaseBean> clases = cd.getDistribucionClasesTree(
```

```
                TreeClassifier, Classlabels, claseIndice);
        j48.setClases(clases);
        // numero de elementos, tamaño del arbol
        int numElementos = cd.getNumElementosArbol(TreeClassifier);
        j48.setNumElementos(numElementos);
        // asginar instancias a sus correspondientes clases
        DatasetBean dataset = cd.asignarInstanciasToClases(TreeClassifier,
                algoritmo);
        dataset.setAtributos(attrs);
        // dataset o conjunto de datos de entrenamiento
        j48.setDataset(dataset);
        return j48;
 }

/**
* Metodo que ejecuta el algrotimo EM (esperanza-maximización) y devuelve
* un objeto de clase EMBean que contiene el resultado del entrenamiento
* de dicho algoritmo utilizando un conjunto de datos, como parametros le
* pasamos el conjunto de datos a utilizar en el entrenamiento y una lista
* con los atributos de dicho conjunto de datos ademas del numero de
* clusteres y el numero maximo de iteraciones.
*
* @param set : conjunto de datos utilizado en el entrenamiento del algoritmo
EM
* @param attrs : lista de atributos del conjunto de datos
* @param numClusters : numero de clusteres
* @param maxIteraciones : el numero maximo de iteraciones necesario hasta
que el
* algoritmo converge
* @return EMBean : objeto que contiene el resultado del entrenamiento
* del algoritmo EM
  * @throws Exception lanza una excepcion generica
  */

 public EMBean EM(DatasetBean set, ArrayList<AtributoBean> attrs,
             int numClusters, int maxIteraciones) throws Exception {

        EMBean em = new EMBean();
        String algoritmo = ClusteringData.ALGORITMO_EM;

        // constriumos el objeto classificador
```

```
        ClusteringData cd = new ClusteringData();
        Object EMClusterer = cd.clustering(set, numClusters, maxIteraciones,
                    algoritmo);
        // almacenamos el numero de clusteres
        em.setNumClusters(cd.getNumeroClusteres(EMClusterer, algoritmo));
        // obtenemos el numero de iteraciones
        em.setNumIteraciones((cd.getNumeroIteraciones(EMClusterer,
algoritmo)));
        // obtenemos el loglikely
        em.setLogLikelihood(cd.getLogLikely(EMClusterer));
        // distribucion de clusteres
        em.setClusters(cd.getDistribucionClusteres(EMClusterer, attrs));
        // asignar las instancias a sus correspondientes clusteres
        DatasetBean dataset = cd.asignarInstanciasToClusteres(EMClusterer,
                    algoritmo);
        dataset.setAtributos(attrs);
        em.setDataset(dataset);
        // asignamos el numero de instancias que pertenecen a cada clusterer
        int numeroClusteres = em.getNumClusters();
        int[] numInstanciasCluster = new int[numeroClusteres];

        // recorremos los clusteres
        for (int i = 0; i < numeroClusteres; i++) {

              ArrayList<InstanciaBean> instancias = dataset.getInstancias();
              // para cada cluster recorremos todas las instancias
              for (int a = 0; a < dataset.getNumeroInstancias(); a++) {
                    if (i == instancias.get(a).getNumCluster())
                          numInstanciasCluster[i]++;
              }
        }
        em.setNumInstanciasCluster(numInstanciasCluster);
        return em
  }

  /**
* Metodo que ejecuta el algrotimo KMeans y devuelve un objeto de clase
* KMeansBean que contiene el resultado del entrenamiento
* de dicho algoritmo utilizando un conjunto de datos, como parametros le
* pasamos el conjunto de datos a utilizar en el entrenamiento y una lista
* con los atributos de dicho conjunto de datos ademas del numero de
```

```
* clusteres y el numero maximo de iteraciones.
*
* @param set  : conjunto de datos utilizado en el entrenamiento del
* algoritmo KMEANS
* @param attrs : lista de atributos del conjunto de datos
* @param numClusters : numero de clusteres
* @param maxIteraciones : el numero maximo de iteraciones necesario hasta
que
* el algoritmo converge
* @return KMeansBean : objeto que contiene el resultado del entrenamiento
* del algoritmo KMEANS
* @throws Exception lanza una excepcion generica
*/

 public KMeansBean KMeans(DatasetBean set, ArrayList<AtributoBean> attrs,
            int numClusters, int maxIteraciones) throws Exception {

       KMeansBean km = new KMeansBean();
       String algoritmo = ClusteringData.ALGORITMO_KMEANS;

       ClusteringData cd = new ClusteringData();
       Object clusterer = cd.clustering(set, numClusters, maxIteraciones,
                   algoritmo);

       // numero centorides o clusteres
       km.setNumClusters(cd.getNumeroClusteres(clusterer, algoritmo));
       // error cuadratico
       km.setSquaredError(cd.getErrorCuadratico(clusterer));
       // numero de iteraciones
       km.setNumIteraciones(cd.getNumeroIteraciones(clusterer, algoritmo));
       // numero de instancias que pertenecen a cada centroide o cluster
       km.setNumInstanciasCluster(cd.getNumInstanciasCluster(clusterer));
       // obtenemos los centroides
       km.setCentroides(cd.getCentroides(clusterer));
       DatasetBean dataset = cd.asignarInstanciasToClusteres(clusterer,
                   algoritmo);
       dataset.setAtributos(attrs);
       km.setDataset(dataset);
       return km;
 }
}
```

Paquete:com.machine.learning.bo

Clase:EvaluarAlgoritmos.java

```
package com.machine.learning.bo;
import java.util.ArrayList;
import com.machine.learning.bean.AtributoBean;
import com.machine.learning.bean.DatasetBean;
import com.machine.learning.bean.EvaluacionClaseBean;
import com.machine.learning.bean.EvaluacionClusterBean;
import com.machine.learning.connectAPI.ClassifierData;
import com.machine.learning.connectAPI.ClusteringData;

public class EvaluarAlgoritmos {

 /**
  * constructor de la clase
  */
 public EvaluarAlgoritmos() {
       super();
 }

 /**
  * Metodo que evalua el algoritmo EM (esperanza-maximización),le pasamos
como
  * parametro un conjunto de datos y nos devuele una lista con objetos de la
  * clase EvaluacionClusterBean que contienen informacion de la evaluacion
de
  * dicho algoritmo
  *
  * @param DatasetBean : conjunto de datos utilizado en la evaluacion del
algoritmo
  * @return ArrayList: un listado de objetos de la clase
EvaluacionClusterBean que contiene
  * el resultado de la evaluacion del algoritmo utilizando el conjunto de
datos pasado
  * como parametro
  * @throws Exception lanza una excepcion generica
  */

 public ArrayList<EvaluacionClusterBean> EvaluarAlgoritmoEM(DatasetBean set)
```

```
            throws Exception {

      ArrayList<EvaluacionClusterBean> listaEval = new
ArrayList<EvaluacionClusterBean>();
      String algoritmo = ClusteringData.ALGORITMO_EM;

      for (int i = 1; i < 11; i++) {

            EvaluacionClusterBean evalBean = new EvaluacionClusterBean();
            ClusteringData cd = new ClusteringData();
            Object clusterer = cd.clustering(set, i, 100, algoritmo);
            Object eval = cd.evalClustering(set, clusterer, algoritmo);
            // numero de instancias
            evalBean.setNumInstancias(set.getNumeroInstancias());
            // numero de clusters
            evalBean.setNumeroClusters(i);
            // log likelihood
            double logLikelihood = cd.getEvalLogLikely(eval);
            evalBean.setLogLikelihood(logLikelihood);
            int[] numInstanciasCluster =
            cd.getNumInstanciasClusterEval(eval);
            evalBean.setNumInstanciasCluster(numInstanciasCluster);
                  // porcentaje del numero de instancias que pertenece a
            //cada uno de
            // los clusteres
            double[] porcentajeInstanciasCluster = new double[evalBean
                        .getNumeroClusters()];
            for (int s = 0; s < evalBean.getNumeroClusters(); s++) {
                  double a = (double) numInstanciasCluster[s]
                              / (double) evalBean.getNumInstancias();
                  porcentajeInstanciasCluster[s] = a * 100.0;
            }

 evalBean.setPorcentajeInstanciasCluster(porcentajeInstanciasCluster);
            // numero de iteraciones
            int numIteraciones = cd.getNumeroIteraciones(clusterer,
algoritmo);
            evalBean.setNumeroIteraciones(numIteraciones);
            // añadimos el objeto de evaluacion evalBean a la lista
            listaEval.add(evalBean);
      }
      return listaEval;
```

```
 }


 /**
  * Metodo que evalua el algoritmo KMEANS ,le pasamos como parametro un
conjunto
  * de datos y nos devuele una lista con objeto de la clase
EvaluacionClusterBean
  * que contienen informacion de evaluacion de dicho algoritmo.
  *
  * @param DatasetBean : conjunto de datos utilizado para evaluar el
algoritmo KMeans
  * @return ArrayList : un listado de objetos de la clase
EvaluacionClusterBean que contiene
  * el resultado de la evaluacion del algoritmo utilizando el conjunto de
datos pasado
  * como parametro.
  * @throws Exception lanza una excepcion generica
  */


 public ArrayList<EvaluacionClusterBean> EvaluarAlgoritmoKMEANS(
             DatasetBean set) throws Exception {
       ArrayList<EvaluacionClusterBean> listaEval = new
ArrayList<EvaluacionClusterBean>();
       String algoritmo = ClusteringData.ALGORITMO_KMEANS;


       for (int i = 1; i < 11; i++) {
             EvaluacionClusterBean evalBean = new EvaluacionClusterBean();
             ClusteringData cd = new ClusteringData();
             Object clusterer = cd.clustering(set, i, 100, algoritmo);
             // numero de instancias
             evalBean.setNumInstancias(set.getNumeroInstancias());
             // suma de erroes cuadraticos
             evalBean.setSumaErrores(cd.getErrorCuadratico(clusterer));
             // numero de instancias
             evalBean.setNumInstancias(set.getNumeroInstancias());
             // numero de iteraciones

 evalBean.setNumeroIteraciones(cd.getNumeroIteraciones(clusterer,
                         algoritmo));
       // numero de instancias que pertenecen a cada centroide o cluster
             int[] numInstanciasCluster =
cd.getNumInstanciasCluster(clusterer);
             evalBean.setNumInstanciasCluster(numInstanciasCluster);
```

```
            // almacenamos el numero de clusters
            evalBean.setNumeroClusters(i);
            // porcentaje instancias cluster
            double[] porcentajeInstanciasCluster = new double[10];
            for (int s = 0; s < numInstanciasCluster.length; s++) {
                  double a = (double) numInstanciasCluster[s]
                              / (double) evalBean.getNumInstancias();
                  porcentajeInstanciasCluster[s] = a * 100.0;
            }

 evalBean.setPorcentajeInstanciasCluster(porcentajeInstanciasCluster);
            listaEval.add(evalBean);
      }
      return listaEval;
 }


 /**
  * Metodo que evalua el algoritmo naive bayes, para ello le pasamos como
parametro
  * un conjunto de datos y el atributo utilizado como clase indice para
clasificar
  * el conjunto de datos.
  *
  * @param DatasetBean : conjunto de datos utilizado para evaluar el
algoritmo Naive Bayes
  * @param claseIndice : atributo utilizado como clase indice o
discriminatoria
  * @return EvaluacionClaseBean : un objeto de la clase
EvaluacionClusterBean que contiene
  * el resultado de la evaluacion del algoritmo utilizando el conjunto de
datos pasado
  * como parametro.
  * @throws Exception lanza una excepcion generica
  *
  */
 public EvaluacionClaseBean EvaluarAlgoritmoNaiveBayes(DatasetBean set,
            int claseIndice) throws Exception {

      EvaluacionClaseBean evalClase = new EvaluacionClaseBean();
      String algoritmo = ClassifierData.ALGORITMO_NAIVE_BAYES;
      evalClase = EvaluarClasificador(set, claseIndice, algoritmo);
      return evalClase;
```

```
 }

/**
* Metodo que evalua el algoritmo  C4.5, para ello le pasamos como parametro
* un conjunto de datos y el atributo utilizado como clase indice para clasificar
* el conjunto de datos.
*
* @param DatasetBean : conjunto de datos utilizado para evaluar el algoritmo Naive Bayes
* @param claseIndice : atributo utilizado como clase indice o discriminatoria
* @return EvaluacionClaseBean : un objeto de la clase EvaluacionClusterBean que contiene
* el resultado de la evaluacion del algoritmo utilizando el conjunto de datos pasado
* como parametro.
* @throws Exception lanza una excepcion generica
*/

 public EvaluacionClaseBean EvaluarAlgoritmoTree(DatasetBean set,
             int claseIndice) throws Exception {
       EvaluacionClaseBean evalClase = new EvaluacionClaseBean();
       String algoritmo = ClassifierData.ALGORITMO_TREE_J48;
       evalClase = EvaluarClasificador(set, claseIndice, algoritmo);
       return evalClase;
 }

 /**
  * Metodo que evalua un algoritmo clasificador (C4.5 y/o Naive Bayes) y devuelve un
* objeto de la clase EvaluacionClaseBean que contiene el resultado de la evaluacion
* del clasificador.
*
* @param DatasetBean : conjunto de datos utilizado para evaluar al clasificador
* @param int : atributo que utilizamos como clase indice o clase discriminatoria
* @param String : algoritmo clasificador que evaluamos
* @return EvaluacionClaseBean : objeto de la clase EvaluacionClaseBean que contiene
* el resultado de la evaluacion
* @throws Exception lanza una excepcion generica
```

```
*/

 private EvaluacionClaseBean EvaluarClasificador(DatasetBean set,
             int claseIndice, String algoritmo) throws Exception {

       EvaluacionClaseBean evalClase = new EvaluacionClaseBean();
       ClassifierData cd = new ClassifierData();
       // algoritmo evaluado
       evalClase.setAlgoritmo(algoritmo);
       // numero de instancias
       int numeroInstancias = set.getNumeroInstancias();
       evalClase.setNumeroInstancias(numeroInstancias);
       // numero de atributos
       int numeroAtributos = set.getNumeroAtributos();
       evalClase.setNumeroAtributos(numeroAtributos);
       Object eval = cd.getEvalClasificador(set, algoritmo, claseIndice);
       // instancias clasificadas correctamente
       double instanciasOK = cd.getInstanciasBienClassificadas(eval);
       evalClase.setInstanciasBienClasificadas(instanciasOK);
       // porcentaje instancias clasificadas correctamente
       double porcentajeOK = instanciasOK / numeroInstancias;
       evalClase.setPorcentajeBienClasificadas(porcentajeOK);
       double precision = porcentajeOK * 100;
       evalClase.setPrecision(precision);
       // instancias clasificadas incorrectamente
       double instanciasKO = cd.getInstanciasMalClassificadas(eval);
       evalClase.setInstanciasMalClasificadas(instanciasKO);
       // porcentaje instancias clasificadas correctamente
       double porcentajeKO = instanciasKO / numeroInstancias;
       evalClase.setPorcentajeMalClasificadas(porcentajeKO);
       // labels clase indice
       ArrayList<AtributoBean> listaAtributos = set.getAtributos();
       AtributoBean attr = (AtributoBean) listaAtributos.get(claseIndice);
       String[] labelsClaseIndice = attr.getPosiblesValores();
       evalClase.setLabelsClaseIndice(labelsClaseIndice);
       // nombre clase indice
       String nombreClaseIndice = attr.getNombre();
       evalClase.setNombreClaseIndice(nombreClaseIndice);
       // matriz de confusion
       double[][] matriz = cd.getMatrizConfusion(eval);
```

```
        evalClase.setMatriz(matriz);
        // coeficiente kappa
        double kappa = cd.getCoeficienteKappa(eval);
        evalClase.setKappa(kappa);
        return evalClase;
 }
}
```

Paquete:com.machine.learning.connectAPI

Clase:ConversorData.java

```
package com.machine.learning.connectAPI;
import java.text.DecimalFormat;
import java.util.ArrayList;
import weka.core.Attribute;
import weka.core.DenseInstance;
import weka.core.Instance;
import weka.core.Instances;
import weka.gui.treevisualizer.Edge;
import weka.gui.treevisualizer.Node;
import com.machine.learning.bean.AtributoBean;
import com.machine.learning.bean.DatasetBean;
import com.machine.learning.bean.InstanciaBean;
import com.machine.learning.bean.NodoArbolBean;

public class ConversorData {

  /**
  * metodo que recoge un Node como parametro que representa un nodo de
  * arbol en la API de WEKA y lo transforma en un objeto de la clase
  * NodoArbolBean que representa un nodo de arbol en la aplicacion
  * @param  n: nodo de arbol de la API de Weka
  * @return NodoArbolBean
  */
 static NodoArbolBean convert_NodoWeka_to_NodoArbol(Node n){

        NodoArbolBean nA = new NodoArbolBean();
```

```
		nA.setLabel(n.getLabel());
		nA.setReferencia(n.getRefer());
		if(n.getParent(0)!=null)
			nA.setRefPadre(n.getParent(0).getRsource());

		ArrayList<NodoArbolBean> hijos = new ArrayList<NodoArbolBean>();
		int i=0;
		if(n.getChild(i)!=null){

			do{
				NodoArbolBean aA = new NodoArbolBean();

				Edge e = n.getChild(i);

				aA = convert_NodoWeka_to_NodoArbol(e.getTarget());
				aA.setArista(nA.getLabel()+e.getLabel());
				hijos.add(aA);
				i++;
			}while(n.getChild(i)!=null);

			nA.setHijos(hijos);
		}
		return nA;
}

 /**
 * metodo para transformar una instancia de la API de WEKA a un objeto de
 * la clase InstanciaBean, que a su vez representa una instancia dentro
 * de la aplicacion.
 * @param  i : instancia de la API de Weka
 * @return InstanciaBean
 */
static InstanciaBean convert_INSTANCE_WEKA_TO_APP(Instance i){

		InstanciaBean  instancia = new InstanciaBean();
		int numeroAttributos = i.numAttributes();
		DecimalFormat df = new DecimalFormat("#0.####");

		for(int j=0;j<numeroAttributos;j++){
			if(i.attribute(j).isNumeric())
```

```
                  instancia.setValue(i.attribute(j).name(),
String.valueOf(df.format(i.value(j))));
            else
                  instancia.setValue(i.attribute(j).name(),
String.valueOf(i.stringValue(j)));
      }
      return instancia;
 }

  /**
  * metodo para transformar un atributo de la API de WEKA a un objeto de la
  * clase AtributoBean, que a su vez representa un atributo dentro
  * de la aplicacion.
  * @param  i : atributo de la API de Weka
  * @return AtributoBean
  */
 static AtributoBean convert_ATTRIBUTE_WEKA_TO_APP(Attribute i){
      AtributoBean  attr = new AtributoBean();
      attr.setNombre(i.name());
      return attr;
 }

  /**
  * metodo que recoge un DatasetBean como parametro que representa un
  * conjunto de datos cargado en la aplicacion, y lo transforma en un
  * conjunto de datos compatible con el API de WEKA.
  * @param DatasetBean : conjunto de datos cargado en la aplicacion.
  * @return conjunto de datos compatible con la API de weka.
  */
 static Instances convert_DATASET_APP_TO_WEKA (DatasetBean set){

      ArrayList<AtributoBean> attrs = set.getAtributos();
      ArrayList<InstanciaBean> instancias = set.getInstancias();
      ArrayList<Attribute> atributos_weka = new ArrayList<Attribute>();
      Instances dataset_weka = new
Instances(set.getNombreDataSet(),atributos_weka ,set.getNumeroInstancias());

      int numero_attrs = attrs.size();
      for(int i=0; i < attrs.size();i++){

            weka.core.Attribute attr_weka = null;
```

```
            AtributoBean attr = attrs.get(i);
            String nombre_attr = attr.getNombre();

            if (attr.getTipo().equals("NUMERIC")) {
                  attr_weka = new Attribute(nombre_attr);
            }else{
                  String[] posiblesValores = attr.getPosiblesValores();

                  ArrayList<String> my_nominal_values = new
ArrayList<String>(posiblesValores.length);
                  for(int j=0;j<posiblesValores.length;j++)
                        my_nominal_values.add(posiblesValores[j]);

                  attr_weka = new Attribute(nombre_attr,
my_nominal_values);
            }

            attr_weka.addRelation(dataset_weka);
            atributos_weka.add(attr_weka);
      }

      dataset_weka = new Instances(set.getNombreDataSet(),atributos_weka
,set.getNumeroInstancias());

for(int k=0; k < instancias.size();k++){

 Instance ins_WEKA = new DenseInstance(numero_attrs);
 ins_WEKA.setDataset(dataset_weka);
 InstanciaBean ins = instancias.get(k);

for(int h=0; h < numero_attrs;h++){
    String dato_aux = ins.getValue(attrs.get(h).getNombre());

if(attrs.get(h).getTipo().equals("NUMERIC")){
if(!dato_aux.equals("?")) ins_WEKA.setValue(h,Double.parseDouble(dato_aux));

                  }else{
                        if(!dato_aux.equals("?"))
ins_WEKA.setValue(h,dato_aux);
                  }
            }
            dataset_weka.add(ins_WEKA);
      }
```

```
        return dataset_weka;
    }
}
```

Paquete:com.machine.learning.connectAPI

Clase:ClassifierData.java

```
package com.machine.learning.connectAPI;
import java.io.StringReader;
import java.lang.reflect.Field;
import java.util.ArrayList;
import java.util.LinkedHashMap;
import com.machine.learning.bean.AtributoBean;
import com.machine.learning.bean.ClaseBean;
import com.machine.learning.bean.DatasetBean;
import com.machine.learning.bean.InstanciaBean;
import com.machine.learning.bean.NodoArbolBean;
import weka.classifiers.bayes.NaiveBayes;
import weka.classifiers.trees.J48;
import weka.classifiers.Evaluation;
import weka.core.Attribute;
import weka.core.Instance;
import weka.core.Instances;
import weka.estimators.DiscreteEstimator;
import weka.estimators.Estimator;
import weka.estimators.NormalEstimator;
import weka.gui.treevisualizer.Node;
import weka.gui.treevisualizer.TreeBuild;

public class ClassifierData {

 public static final String ALGORITMO_NAIVE_BAYES = "NB";
 public static final String ALGORITMO_TREE_J48 = "C4.5 TREE";
 public static final int ALGORITMO_NO_SOPORTADO = -1;
 private Instances dataset;

 /**
  * Constructor por defecto
  */
 public ClassifierData() {
```

```
        super();
 }


/**
* metodo que entrena un algoritmo de clasificacion y almacena el resultado
de
* dicho entrenamiento en un objeto clasificador, como parametros le pasamos
el conjunto
* de datos que sera utilizado en el entrenamiento del agloritmo , el nombre
del algoritmo
* y el atributo utilizado como clase indice
* @param set : conjunto de datos que sera utilizado en el entrenamiento del
algoritmo
* @param claseIndice : atributo utilizado como clase indice o
discriminatoria
* @param algoritmo : algoritmo a entrenar con el conjunto de datos dado
* @return objeto clasificador que contiene el resultado del entrenamiento
del algoritmo
* @throws Exception lanza una excepcion generica
*/
 public Object Classifier(DatasetBean set, int claseIndice, String
algoritmo)
              throws Exception {


        Instances datasetWeka =
ConversorData.convert_DATASET_APP_TO_WEKA(set);
        this.dataset = datasetWeka;


        if (algoritmo.equals(ClassifierData.ALGORITMO_NAIVE_BAYES)) {


              NaiveBayes nb = new NaiveBayes();
              datasetWeka.setClassIndex(claseIndice);
              nb.buildClassifier(datasetWeka);
              return nb;


        } else if (algoritmo.equals(ClassifierData.ALGORITMO_TREE_J48)) {


              J48 tree = new J48();
              datasetWeka.setClassIndex(claseIndice);
              tree.buildClassifier(datasetWeka);
              return tree;
        } else {
              return ClassifierData.ALGORITMO_NO_SOPORTADO;
```

```
	}
 }


 /**
  * devuelve el numero de elementos de un arbol del algoritmo C4.5
  * @param Classifier : objeto clasificador que almacena la informacion del
entrenamiento
  * de un algoritmo de clasificacion
  * @return numero de elmentos de un arbol
  * @throws Exception lanza una excepcion generica
  */
 public int getNumElementosArbol(Object Classifier) throws Exception {

	J48 tree = (J48) Object.class.cast(Classifier);
	int numElementos = tree.numElements();
	return numElementos;
 }


 /**
  * metodo que devuelve un string en formato DOT (graph description
language)
  * que describe graficamente el arbol a traves de texto
  * @param Classifier : objeto clasificador que almacena la informacion del
entrenamiento
  * de un algoritmo de clasificacion
  * @return cadena de texto que describe el arbol
  * @throws Exception lanza una excepcion generica
  */
 public String getDotGraph(Object Classifier) throws Exception {
	J48 tree = (J48) Object.class.cast(Classifier);
	String graph = tree.graph();
	return graph;
 }


/**
* metodo que devuelve un objeto de la clase NodoArbolBean que contiene un
arbol
* obtenido como resultado de entrenar el algoritmo c4.5 con un conjunto de
datos dado.
* @param Classifier : objeto clasificador que almacena la informacion del
entrenamiento
* de un algoritmo de clasificacion
```

```
* @return un arbol c4.5 con todos sus componentes
* @throws Exception lanza una excepcion generica
*/
 public NodoArbolBean getArbol(Object Classifier) throws Exception {

       J48 tree = (J48) Object.class.cast(Classifier);
       String graph = tree.graph();
       TreeBuild tb = new TreeBuild();
       StringReader st = new StringReader(graph);
       Node n = tb.create(st);
       NodoArbolBean arbol = ConversorData.convert_NodoWeka_to_NodoArbol(n);
       return arbol;

 }

/**
* metodo que devuelve el numero de instancias suavizado, el numero de
instancias
* suavizado es el numero de instancias despues de aplicar la estimacion de
laplace
* como tecnica de suavizado (smoothing technique)
* @param Classifier : objeto clasificador que almacena la informacion del
entrenamiento
* de un algoritmo de clasificacion
* @return numero de instancias suavizado
* @throws Exception lanza una excepcion generica
*/
 public double getNumInstanciasSuavizado(Object Classifier) throws Exception
{

       Field privateStringField3 =
Classifier.getClass().getDeclaredField("m_ClassDistribution");
       privateStringField3.setAccessible(true);
       Estimator m_class = (Estimator) privateStringField3.get(Classifier);
       DiscreteEstimator discreteEstimator = (DiscreteEstimator) m_class;
       double numInstanciasSuavizado = discreteEstimator.getSumOfCounts();
       return numInstanciasSuavizado;
 }

/**
* metodo que devuelve un conjunto de datos que contiene un listado de todas
las
```

```
* instancias, cada una de dichas instancias sera asignada a su
correspondiente clase
* @param Classifier : objeto clasificador que almacena la informacion del
*entrenamiento de un algoritmo de clasificacion
* @param algoritmo : algoritmo que fue utilizado en el entrenamiento
* @return conjunto de datos
* @throws Exception lanza una excepcion generica
*/
 public DatasetBean asignarInstanciasToClases(Object Classifier,
            String algoritmo) throws Exception {

      Instances datasetWeka = this.dataset;
      DatasetBean dataset = new DatasetBean();

      if (algoritmo.equals(ClassifierData.ALGORITMO_TREE_J48)) {

            J48 tree = (J48) Object.class.cast(Classifier);
            for (int i = 0; i < datasetWeka.numInstances(); i++) {
                  Instance instanciaWeka = datasetWeka.instance(i);
                  InstanciaBean instancia = ConversorData
                              .convert_INSTANCE_WEKA_TO_APP(instanciaWeka);
                  Double e = tree.classifyInstance(instanciaWeka);
                  instancia.setNumClase(e.intValue());
                  dataset.addInstancia(instancia);
            }

      } else if (algoritmo.equals(ClassifierData.ALGORITMO_NAIVE_BAYES)) {

            NaiveBayes nb = (NaiveBayes) Object.class.cast(Classifier);
            for (int i = 0; i < datasetWeka.numInstances(); i++) {
                  Instance instanciaWeka = datasetWeka.instance(i);
                  InstanciaBean instancia = ConversorData
                              .convert_INSTANCE_WEKA_TO_APP(instanciaWeka);
                  Double e = nb.classifyInstance(instanciaWeka);
                  instancia.setNumClase(e.intValue());
                  dataset.addInstancia(instancia);
            }
      }
      return dataset;
 }
```

```
/**
* metodo que calcula la distribucion de clases, devuelve un listado
* de clases con toda la informacion obtenida del entrenamiento del
* algoritmo de clasificacion C4.5
* @param Classifier : objeto clasificador que almacena la informacion
* del entrenamiento de un algoritmo C4.5
* @param Classlabels :array que contiene etiquetas descriptivas para
* cada una de las clases
 * @param claseIndice : atributo utilizado como clase indice o
discriminatoria
* @return listado de clases
* @throws Exception lanza una excepcion generica
*/
 public ArrayList<ClaseBean> getDistribucionClasesTree(Object Classifier,
            String[] Classlabels, int claseIndice) throws Exception {

       ArrayList<ClaseBean> listaClases = new ArrayList<ClaseBean>();

       J48 tree = (J48) Object.class.cast(Classifier);
       int numeroClases = this.dataset.numClasses();
       double[] numInstanciasClase = new double[numeroClases];
       int numeroInstancias = this.dataset.numInstances();

       for (int y = 0; y < numeroClases; y++) {

             ClaseBean clase = new ClaseBean();
             for (int x = 0; x < numeroInstancias; x++) {
                   Instance ins = this.dataset.instance(x);
                   double distro = tree.classifyInstance(ins);
                   if (distro == y)
                         numInstanciasClase[y]++;
             }
             clase.setLabel(Classlabels[y]);
             clase.setNumeroInstancias(numInstanciasClase[y]);
             clase.setIndice(y);
             listaClases.add(clase);
       }
       return listaClases;
 }

/**
```

```
* metodo que calcula la distribucion de clases, devuelve un listado
* de clases con toda la informacion obtenida del entrenamiento de un
* algoritmo de clasificacion
* @param Classifier : objeto clasificador que almacena la informacion
* del entrenamiento de un algoritmo de clasificacion
* @param Classlabels : array que contiene etiquetas descriptivas para
* cada una de las clases
* @param claseIndice : atributo utilizado como clase indice o
discriminatoria
* @return listado de clases
* @throws Exception lanza una excepcion generica
*/
 public ArrayList<ClaseBean> getDistribucionClases(Object Classifier,
            String[] Classlabels, int claseIndice) throws Exception {

      // distribucion clases
      Field privateStringField2 = Classifier.getClass().getDeclaredField(
                  "m_Distributions");
      privateStringField2.setAccessible(true);
      Estimator[][] m_model = (Estimator[][]) privateStringField2
                  .get(Classifier);

      // probabilidad de clase
      Field privateStringField3 = Classifier.getClass().getDeclaredField(
                  "m_ClassDistribution");
      privateStringField3.setAccessible(true);
      Estimator m_class = (Estimator) privateStringField3.get(Classifier);
      DiscreteEstimator discreteEstimator = (DiscreteEstimator) m_class;
      double numInstanciasSuavizado = discreteEstimator.getSumOfCounts();

      int numElementos = discreteEstimator.getNumSymbols();
      double[] probabilidadClase = new double[numElementos];
      for (int i = 0; i < numElementos; i++) {
            probabilidadClase[i] = discreteEstimator.getCount(i);
      }

      // buscamos el atributo classindex y lo quitamos de la lista de
      // atributos
      ArrayList<Attribute> AtributosWeka = new ArrayList<Attribute>();
      for (int f = 0; f < this.dataset.numAttributes(); f++) {
            if (f != claseIndice)
```

```
                AtributosWeka.add(this.dataset.attribute(f));
        }

        ArrayList<ClaseBean> listaClases = new ArrayList<ClaseBean>();

        // recorremos las clases, una iteracion por cada clase
        for (int j = 0; j < this.dataset.numClasses(); j++) {

              ClaseBean clase = new ClaseBean();
              ArrayList<AtributoBean> atributos = new
ArrayList<AtributoBean>();

              // recorremos todos los atributos para cada una de las clases
              for (int k = 0; k < AtributosWeka.size(); k++) {

                    AtributoBean atributo = new AtributoBean();
                    Attribute attrWeka = AtributosWeka.get(k);

                    if (attrWeka.isNumeric()) {

                    NormalEstimator e = (NormalEstimator) m_model[k][j];
                          atributo.setNombre(attrWeka.name());
                          atributo.setTipo("NUMERIC");
                          atributo.setMedia(e.getMean());
                          atributo.setStdDev(e.getStdDev());

                    } else if (attrWeka.isNominal()) {

                          atributo.setTipo("NOMINAL");
                    DiscreteEstimator e = (DiscreteEstimator) m_model[k][j];
                          atributo.setNombre(attrWeka.name());
                                LinkedHashMap<String, String> labels = new
                    LinkedHashMap<String, String>();

                          for (int z = 0; z < attrWeka.numValues(); z++)
                                labels.put(attrWeka.value(z),
                                            String.valueOf(e.getCount(z)));

                          atributo.setLabels(labels);
                    }
                    atributos.add(atributo);
```

```
            }

            clase.setLabel(Classlabels[j]);
            clase.setNumeroInstancias(probabilidadClase[j]);
            clase.setIndice(j);
            clase.setNumInstanciasSuavizado(numInstanciasSuavizado);
            clase.setAtributos(atributos);
            listaClases.add(clase);
      }
      return listaClases;
 }

/**
* metodo que evalua un algoritmo de clasifiacion y almacena el resultado de
* dicha evaluacion en un objeto evaluador, como parametros le pasamos el
conjunto
* de datos que sera utilizado en la evaluacion del agloritmo , el nombre del
* algoritmo y el atributo utilizado como clase indice
* @param set : conjunto de datos que sera utilizado en la evaluacion
* @param algoritmo : algoritmo que sera evaluado
* @param claseIndice : atributo utilizado como clase indice o
discriminatoria
* @return un objeto evaluador que almacena el resultado de la evaluacion
* de un algoritmo de clasificacion
* @throws Exception lanza una excepcion generica
*/
 public Object getEvalClasificador(DatasetBean set, String algoritmo,
            int claseIndice) throws Exception {

      // convertimos dataset de APP -->> a dataset de WEKA

      Instances datasetWeka =
ConversorData.convert_DATASET_APP_TO_WEKA(set);
      this.dataset = datasetWeka;

      datasetWeka.setClassIndex(claseIndice);
      // crear un objeto de evaluacion
      Evaluation eTest = new Evaluation(datasetWeka);
      if (algoritmo.equals(ClassifierData.ALGORITMO_NAIVE_BAYES)) {

            NaiveBayes nb = new NaiveBayes();
            nb.buildClassifier(datasetWeka);
```

```
            eTest.evaluateModel(nb, datasetWeka);
            return eTest;

      } else if (algoritmo.equals(ClassifierData.ALGORITMO_TREE_J48)) {

            J48 tree = new J48();
            tree.buildClassifier(datasetWeka);
            eTest.evaluateModel(tree, datasetWeka);
            return eTest;
      } else {
            return ClassifierData.ALGORITMO_NO_SOPORTADO;
      }
 }

/**
* metodo que devuelve el numero de instancias classificadas correctamente
* @param eval : objeto evaluador que almacena la informacion de la
evaluacion de
* un algoritmo de clasificacion
* @return numero de instancias classificadas correctamente
* @throws Exception lanza una excepcion generica
*/
 public double getInstanciasBienClassificadas(Object eval) throws Exception
{
      Evaluation eTest = (Evaluation) Object.class.cast(eval);
      return eTest.correct(); }
/**
* metodo que devuelve el numero de instancias classificadas incorrectamente
* @param eval : objeto evaluador que almacena la informacion de la
evaluacion de
* un algoritmo de clasificacion
  * @return numero de instancias classificadas incorrectamente
  * @throws Exception lanza una excepcion generica
  */
 public double getInstanciasMalClassificadas(Object eval) throws Exception {
      Evaluation eTest = (Evaluation) Object.class.cast(eval);
      return eTest.incorrect();
 }

/**
* devuelve el coeficiente Kappa
```

```
* @param eval : objeto evaluador que almacena la informacion de la
evaluacion  de
* un algoritmo de clasificacion
* @return coeficiente Kappa
* @throws Exception lanza una excepcion generica
*/
 public double getCoeficienteKappa(Object eval) throws Exception {
       Evaluation eTest = (Evaluation) Object.class.cast(eval);
       return eTest.kappa();
 }

/**
* metodo devuelve un array de dos dimensiones que contiene la matriz de
confusion
* @param eval : objeto evaluador que almacena la informacion de la
evaluacion de
* un algoritmo de clasificacion
* @return array de dos dimensiones que contiene la matriz de confusion
* @throws Exception lanza una excepcion generica
*/
 public double[][] getMatrizConfusion(Object eval) throws Exception {
       Evaluation eTest = (Evaluation) Object.class.cast(eval);
       double[][] cmMatrix = eTest.confusionMatrix();
       return cmMatrix; }
}
```

Paquete:com.machine.learning.connectAPI

Clase:ClusteringData.java

```
 package com.machine.learning.connectAPI;
import java.lang.reflect.Field;
import java.util.ArrayList;
import java.util.LinkedHashMap;
import com.machine.learning.bean.AtributoBean;
import com.machine.learning.bean.ClusterBean;
import com.machine.learning.bean.DatasetBean;
import com.machine.learning.bean.InstanciaBean;
import weka.clusterers.ClusterEvaluation;
import weka.clusterers.EM;
import weka.clusterers.SimpleKMeans;
import weka.core.Attribute;
```

```
import weka.core.Instance;
import weka.core.Instances;
import weka.estimators.DiscreteEstimator;
import weka.estimators.Estimator;

public class ClusteringData {

 public static final String ALGORITMO_KMEANS = "KM";
 public static final String ALGORITMO_EM = "EM";
 public static final int ALGORITMO_NO_SOPORTADO = -1;
 private Instances dataset;

 /**
  * Constructor por defecto
  */
 public ClusteringData() {
       super();
 }

 /**
  * metodo que entrena un algoritmo de agrupamiento(clustering) y almacena
el resultado de
  * dicho entrenamiento en un objeto agrupador, como parametros le pasamos
el conjunto
  * de datos que sera utilizado en el entrenamiento del agloritmo , el
nombre del algoritmo,
  * el numero de clusteres y el numero maximo de iteraciones
*
* @param set : conjunto de datos que sera utilizado en el entrenamiento del
algoritmo
* @param numClusters : numero de clusteres o grupos que obtenemos como
resultado
* @param maxIteraciones : el numero maximo permitido de iteraciones, el
algoritmo
* no puede ejecutarse mas veces que dicho numero
* @param algoritmo : algoritmo que sera entrenado
* @return objeto agrupador que contiene el resultado del entrenamiento del
algoritmo
* @throws Exception lanza una excepcion generica
*
*/

 public Object clustering(DatasetBean set, int numClusters,
```

```
            int maxIteraciones, String algoritmo) throws Exception {

      Instances datasetWeka =
ConversorData.convert_DATASET_APP_TO_WEKA(set);
      this.dataset = datasetWeka;

      if (algoritmo.equals(ClusteringData.ALGORITMO_KMEANS)) {

            SimpleKMeans clusterer = new SimpleKMeans();
            clusterer.setNumClusters(numClusters);
            clusterer.setMaxIterations(maxIteraciones);
            clusterer.buildClusterer(datasetWeka);
            return clusterer;

      } else if (algoritmo.equals(ClusteringData.ALGORITMO_EM)) {

            EM clusterer = new EM();
            clusterer.setMaxIterations(maxIteraciones);
            clusterer.setNumClusters(numClusters);
            clusterer.buildClusterer(datasetWeka);
            return clusterer;

      } else {

            return ClusteringData.ALGORITMO_NO_SOPORTADO;
      }
 }

/**
* metodo que evalua un algoritmo de agrupamiento(clustering) y almacena el
resultado de
* dicha evaluacion en un objeto evaluador, como parametros le pasamos el
conjunto
* de datos que sera utilizado en la evaluacion del agloritmo , un objeto
agrupador y
* el algoritmo que sera evaluado
* @param set : conjunto de datos que sera utilizado en la evaluacion del
algoritmo
* @param clusterer : objeto agrupador que almacena la informacion del
entrenamiento
* de un algoritmo de agrupamiento (clustering)
* @param algoritmo : algoritmo utilizado en la agrupacion(clustering)
```

```
* @return objeto evaluador que contiene el resultado de la evaluacion del
algoritmo
* @throws Exception lanza una excepcion generica
*/
 public Object evalClustering(DatasetBean set, Object clusterer,
             String algoritmo) throws Exception {

       Instances datasetWeka =
ConversorData.convert_DATASET_APP_TO_WEKA(set);
       this.dataset = datasetWeka;
       ClusterEvaluation eval = new ClusterEvaluation();

       if (algoritmo.equals(ClusteringData.ALGORITMO_KMEANS)) {
             SimpleKMeans skm = (SimpleKMeans) Object.class.cast(clusterer);
             eval.setClusterer(skm);
             eval.evaluateClusterer(datasetWeka);
             return eval;

       } else if (algoritmo.equals(ClusteringData.ALGORITMO_EM)) {
             EM em = (EM) Object.class.cast(clusterer);
             eval.setClusterer(em);
             eval.evaluateClusterer(datasetWeka);
             return eval;

       } else {
             return ClusteringData.ALGORITMO_NO_SOPORTADO;
       }
 }



 /**
* metodo que devuelve el numero de clusteres en los cuales se dividen las
instancias
* del conjunto de datos utilizado en el entrenamiento del algoritmo
* de agrupacion (clustering)
* @param clusterer : objeto agrupador que almacena la informacion del
entrenamiento
* de un algoritmo de agrupamiento (clustering)
* @param algoritmo : algoritmo utilizado en la agrupacion(clustering)
* @return numero de clusteres
* @throws Exception lanza una excepcion generica
*/
```

```
 public int getNumeroClusteres(Object clusterer, String algoritmo)
             throws Exception {

       if (algoritmo.equals(ClusteringData.ALGORITMO_KMEANS)) {
             SimpleKMeans skm = (SimpleKMeans) Object.class.cast(clusterer);
             return skm.getNumClusters();

       } else if (algoritmo.equals(ClusteringData.ALGORITMO_EM)) {
             EM em = (EM) Object.class.cast(clusterer);
             return em.numberOfClusters();
       } else {
             return ClusteringData.ALGORITMO_NO_SOPORTADO;
       }
 }

/**
* metodo que devuele el error cuadratico dentro del cluster para un
algoritmo
* de agrupamiento(clustering)
* @param clusterer : objeto que almacena la informacion del entrenamiento de
* un algoritmo de agrupamiento(clustering)
* @return error cuadratico dentro del cluster
* @throws Exception lanza una excepcion generica
*/
 public double getErrorCuadratico(Object clusterer) throws Exception {

       SimpleKMeans c = (SimpleKMeans) Object.class.cast(clusterer);
       return c.getSquaredError();
 }

 /**
* metodo que devuelve el numero de iteraciones optimo para que un algoritmo
* de agrupamiento(clustering) converge
* @param clusterer : objeto que almacena la informacion del entrenamiento de
* un algoritmo de agrupamiento(clustering)
* @param algoritmo : algoritmo utilizado en el entrenamiento
* @return numero de iteraciones optimo
* @throws Exception lanza una excepcion generica
*/
 public int getNumeroIteraciones(Object clusterer, String algoritmo)
             throws Exception {
```

```
	if (algoritmo.equals(ClusteringData.ALGORITMO_KMEANS)) {
		Field private_m_Iterations = 
clusterer.getClass().getDeclaredField("m_Iterations");
		private_m_Iterations.setAccessible(true);
		int num_Iterations = (int) private_m_Iterations.get(clusterer);
		return num_Iterations;

	} else if (algoritmo.equals(ClusteringData.ALGORITMO_EM)) {

		Field private_m_iterationsPerformed = 
clusterer.getClass().getDeclaredField("m_iterationsPerformed");
		private_m_iterationsPerformed.setAccessible(true);
		int numIteraciones = (int) private_m_iterationsPerformed
				.get(clusterer);
		return numIteraciones;
	} else {

		return ClusteringData.ALGORITMO_NO_SOPORTADO;
	}
 }

 /**
* devuelve un array que contiene el numero de instancias que corresponden a 
cada
* uno de los clusteres obtenidos como resultado del entrenamiento
* de un algoritmo de agrupamiento(clustering)
* @param clusterer : objeto que almacena la informacion del entrenamiento de
* un algoritmo de agrupamiento(clustering)
* @return numero de instancias para cada uno de los clusteres generados
* @throws Exception lanza una excepcion generica
*/
 public int[] getNumInstanciasCluster(Object clusterer) throws Exception {

	// obtenemos el numero de instancias dentro de cada cluster
	Field private_m_ClusterSizes = 
	clusterer.getClass().getDeclaredField("m_ClusterSizes");
	private_m_ClusterSizes.setAccessible(true);
	int[] m_ClusterSizes = (int[]) private_m_ClusterSizes.get(clusterer);
	return m_ClusterSizes;
 }
```

```
/**
* devuelve un array que contiene el numero de instancias que corresponden a
cada
* uno de los clusteres obtenidos como resultado de la evaluacion
* de un algoritmo de agrupamiento(clustering)
* @param eval :objeto evaluador que almacena la informacion de la evaluacion
de
* un algoritmo de agrupamiento(clustering)
* @return numero de instancias para cada uno de los clusteres generados
* @throws Exception lanza una excepcion generica
*/
 public int[] getNumInstanciasClusterEval(Object eval) throws Exception {

       // obtenemos el numero de instancias dentro de cada cluster
       ClusterEvaluation e = (ClusterEvaluation) Object.class.cast(eval);
       double clusterAssignments[] = e.getClusterAssignments();
       int numeroClusteres = e.getNumClusters();
       // numero instancias que pertence a cada uno de los clusteres
       int[] numInstanciasCluster = new int[numeroClusteres];
       for (int y = 0; y < numeroClusteres; y++)
             for (int x = 0; x < clusterAssignments.length; x++)
                   if (clusterAssignments[x] == y)
                         numInstanciasCluster[y]++;
       return numInstanciasCluster;
 }

/**
* metodo que devuelve los centroides obtenidos como resultado del
entrenamiento
* de un algoritmo de agrupacion (clustering)
* @param clusterer : objeto que almacena la informacion del entrenamiento de
* un algoritmo de agrupamiento(clustering)
* @return los centroides obtenidos como resultado del entrenamiento
* de un algoritmo de agrupacion
* @throws Exception lanza una excepcion generica
*/
 public ArrayList<InstanciaBean> getCentroides(Object clusterer)
             throws Exception {
       SimpleKMeans skm = (SimpleKMeans) Object.class.cast(clusterer);
       Instances centroides = skm.getClusterCentroids();
       ArrayList<InstanciaBean> listaCentroides = new
ArrayList<InstanciaBean>();
```

```
	for (int i = 0; i < skm.numberOfClusters(); i++) {
		Instance centroideWeka = centroides.instance(i);
		InstanciaBean centroide = ConversorData
				.convert_INSTANCE_WEKA_TO_APP(centroideWeka);
		listaCentroides.add(centroide);
	}

	return listaCentroides;
 }

/**
* metodo que asigna cada una de las instancias  a su correspondiente cluster
* @param clusterer : objeto que almacena la informacion del entrenamiento de
* un algoritmo de agrupamiento(clustering)
* @param algoritmo : algoritmo que fue utilizado en el entrenamiento
* @return conjunto de datos
* @throws Exception lanza una excepcion generica
*/
 public DatasetBean asignarInstanciasToClusteres(Object clusterer,
		String algoritmo) throws Exception {

	Instances datasetWeka = this.dataset;
	DatasetBean dataset = new DatasetBean();

	if (algoritmo.equals(ClusteringData.ALGORITMO_KMEANS)) {

		SimpleKMeans skm = (SimpleKMeans) Object.class.cast(clusterer);

		for (int i = 0; i < datasetWeka.numInstances(); i++) {
			Instance instanciaWeka = datasetWeka.instance(i);
			InstanciaBean instancia = ConversorData
					.convert_INSTANCE_WEKA_TO_APP(instanciaWeka);

 instancia.setNumCluster(skm.clusterInstance(instanciaWeka));
			dataset.addInstancia(instancia);
		}

	} else if (algoritmo.equals(ClusteringData.ALGORITMO_EM)) {

		EM em = (EM) Object.class.cast(clusterer);
```

```
            for (int i = 0; i < datasetWeka.numInstances(); i++) {
                  Instance instanciaWeka = datasetWeka.instance(i);
                  InstanciaBean instancia = ConversorData
                              .convert_INSTANCE_WEKA_TO_APP(instanciaWeka);

 instancia.setNumCluster(em.clusterInstance(instanciaWeka));
                  dataset.addInstancia(instancia);
            }
      }
      return dataset;
 }

/**
* metodo que calcula la distribucion de clusteres, devuelve un listado
* de clusteres con toda la informacion obtenida del entrenamiento de
* un algoritmo de agrupamiento (clustering)
* @param clusterer : objeto que almacena la informacion del entrenamiento de
* un algoritmo de agrupamiento(clustering)
* @param attrs : listado de atributos del conjunto de datos utilizado en el
* entrenamiento del algoritmo de agrupamiento(clustering)
* @return listado de clusteres
* @throws Exception lanza una excepcion generica
*/
 public ArrayList<ClusterBean> getDistribucionClusteres(Object clusterer,
            ArrayList<AtributoBean> attrs) throws Exception {

      EM em = (EM) Object.class.cast(clusterer);
      ArrayList<ClusterBean> clusteresEM = new ArrayList<ClusterBean>();

      // probablidad de cada cluster
      double[] probablidadCluster = em.clusterPriors();

      // modeloNumerico
      // [cluster][Atributo][0-media,1-desvEsta,2-numinstancias]
      double[][][] modeloNumerico = em.getClusterModelsNumericAtts();

      Field privateStringField =
clusterer.getClass().getDeclaredField("m_model");
      privateStringField.setAccessible(true);
```

```
        Estimator[][] m_model = (Estimator[][])
privateStringField.get(clusterer);

        // clusteres
        for (int j = 0; j < em.numberOfClusters(); j++) {

              ClusterBean cluster = new ClusterBean();
              ArrayList<AtributoBean> atributos = new
ArrayList<AtributoBean>();

              Instances datasetWeka = this.dataset;

              // atributos
              for (int k = 0; k < attrs.size(); k++) {

                    AtributoBean atributo = new AtributoBean();
                    Attribute attrWeka = datasetWeka.attribute(k);

                    if (attrWeka.isNumeric()) {
                          atributo.setNombre(attrWeka.name());
                          atributo.setTipo("NUMERIC");
                          atributo.setMedia(modeloNumerico[j][k][0]);
                          atributo.setStdDev(modeloNumerico[j][k][1]);

                    } else if (attrWeka.isNominal()) {
                          atributo.setTipo("NOMINAL");
                          DiscreteEstimator e = (DiscreteEstimator)
                          m_model[j][k];
                          atributo.setNombre(attrWeka.name());
                          LinkedHashMap<String, String> labels =
                                 new LinkedHashMap<String, String>();
                          for (int z = 0; z < attrWeka.numValues(); z++)
                                labels.put(attrWeka.value(z),
                                            String.valueOf(e.getCount(z)));

                          atributo.setLabels(labels);
                    }

                    atributos.add(atributo);
              }
```

```
            cluster.setProbablidad(probablidadCluster[j]);
            cluster.setIndice(j);
            cluster.setAtributos(atributos);
            clusteresEM.add(cluster);
      }
      return clusteresEM;
 }

 /**
  * calcula el coeficiente de verosimilitud para el objeto clusterer
  * pasado como parametro
  * @param clusterer : objeto que almacena la informacion del entrenamiento
de
  * un algoritmo de agrupamiento(clustering)
  * @return coeficiente de verosimilitud
  * @throws Exception lanza una excepcion generica
  */
 public double getLogLikely(Object clusterer) throws Exception {

      EM em = (EM) Object.class.cast(clusterer);
      Field private_m_loglikely = em.getClass().getDeclaredField(
                  "m_loglikely");
      private_m_loglikely.setAccessible(true);
      double logLikelihood = (double) private_m_loglikely.get(em);
      return logLikelihood;
 }

/**
* calcula el coeficiente de verosimilitud para el objeto eval pasado
* como parametro
* @param eval : objeto evaluador que almacena la información
* de la evaluacion de  un algoritmo de agrupamiento(clustering)
* @return coeficiente de verosimilitud
* @throws Exception lanza una excepcion generica
*/
 public double getEvalLogLikely(Object eval) throws Exception {
      ClusterEvaluation e = (ClusterEvaluation) Object.class.cast(eval);
      double logLikelihood = e.getLogLikelihood();
      return logLikelihood;
 }
}
```