# Práctica 5: Deployment of a Web Information System with data persistence byusing Docker containers

Grado en Ingeniería Informática (Plan 439)

Sistemas de información
Curso: 2025-2026

Sergio Saura Oliva (838585)
José Secadura Del Olmo (815327)
Chloé Bolle-Reddat (961562)

**Fecha de entrega: 27/11/2025**

Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

**Universidad de Zaragoza - Escuela de Ingeniería y Arquitectura**

# Índice

# 1. Introducción

En esta práctica vamos a utilizar Apache Lucene para desarrollar una aplicación capaz de buscar términos dentro de un directorio que contiene más de un centenar de ficheros.

Lucene es una librería diseñada para la recuperación de información, cuyo funcionamiento se basa en la creación de un índice invertido usando varias clases fundamentales. Este tipo de índice permite saber en qué documentos aparece cada término. Durante la indexación y la búsqueda interviene un Analyzer que procesa el texto para obtener su representación interna: separación en tokens, conversión a minúsculas, eliminación de stopwords, etc.

Una vez creado el índice, las búsquedas se realizan a través de IndexSearcher, que ejecuta consultas representadas mediante objetos Query. El resultado de cada consulta se materializa en un objeto Hit asociado a un documento del índice. Estos resultados incluyen un score, que cuantifica la relevancia del documento respecto a la consulta realizada.

# 2. Prueba de Lucene

Primero, probemos Lucene con diferentes analizadores y opciones. Los resultados para cada analizador están abajo, incluyendo el número de hits, el score y los archivos correspondientes.

Antes de todo, observábamos que la re-indexación de todos los archivos a cada ejecución es necesaria. En efecto, como podemos ver en el ejemplo abajo, encontramos los mismos archivos varias veces aunque debe ser único. Eso es porque, cuando reutilizamos el mismo índice, los resultados de cada ejecución se almacenan, lo que falsifica los resultados.

```
Buscando Contaminación: Encontrados 4 hits.
1. ./ficheros/uno.txt   1.8207742
2. ./ficheros/uno.txt   1.8207742
3. ./ficheros/uno.txt   1.8207742
4. ./ficheros/uno.txt   1.8207742

Buscando cambio climático: Encontrados 4 hits.
1. ./ficheros/cuatro.txt        1.1433531
2. ./ficheros/cuatro.txt        1.1433531
3. ./ficheros/cuatro.txt        1.1433531
4. ./ficheros/cuatro.txt        1.1433531
```

Para evitar este problema, forzamos la re-indexación usando el modo de apertura CREATE del IndexWriter. Ahora, con este modo probamos los diferentes analizadores.

Con el simpleAnalizer nos quedó:

```
Buscando Contaminación: Encontrados 1 hits.
1. ./ficheros/uno.txt   1.6400275

Buscando Contaminacion: Encontrados 0 hits.

Buscando cambio climático: Encontrados 3 hits.
1. ./ficheros/cuatro.txt        1.3217045
2. ./ficheros/tres.txt  1.0441608
3. ./ficheros/dos.txt   0.94759953

Buscando cambio climatico: Encontrados 3 hits.
1. ./ficheros/cuatro.txt        0.66085225
2. ./ficheros/dos.txt   0.53812945
3. ./ficheros/tres.txt  0.5220804

Buscando cambio: Encontrados 3 hits.
1. ./ficheros/cuatro.txt        0.66085225
2. ./ficheros/dos.txt   0.53812945
3. ./ficheros/tres.txt  0.5220804

Buscando climático: Encontrados 3 hits.
1. ./ficheros/cuatro.txt        0.6554797
2. ./ficheros/tres.txt  0.51912993
3. ./ficheros/dos.txt   0.41299203

Buscando climatico: Encontrados 0 hits.

Buscando por: Encontrados 4 hits.
1. ./ficheros/uno.txt   0.20090158
2. ./ficheros/tres.txt  0.19296822
3. ./ficheros/cuatro.txt        0.19090943
4. ./ficheros/dos.txt   0.15896153

Buscando aeropuerto: Encontrados 1 hits.
1. ./ficheros/uno.txt   2.3586044
```

## 2.1. Preguntas sobre Lucene

Las siguientes son las cuestiones a investigar y documentar sobre el comportamiento de Lucene:

- What happens if we use the StandardAnalyzer instead of the SimpleAnalyzer? What role does the stopwords.txt file play?

  La salida que se ha obtenido usando el StandardAnalyzer es:

  ```
  Buscando Contaminación: Encontrados 1 hits.
  ```

```
1. ./ficheros/uno.txt   1.6837957

Buscando Contaminacion: Encontrados 0 hits.

Buscando cambio climatico: Encontrados 3 hits.
1. ./ficheros/cuatro.txt        0.6554797
2. ./ficheros/dos.txt   0.54116195
3. ./ficheros/tres.txt  0.51912993

Buscando cambio climático: Encontrados 3 hits.
1. ./ficheros/cuatro.txt        1.3109595
2. ./ficheros/tres.txt  1.0382599
3. ./ficheros/dos.txt   0.954154

Buscando cambio: Encontrados 3 hits.
1. ./ficheros/cuatro.txt        0.6554797
2. ./ficheros/dos.txt   0.54116195
3. ./ficheros/tres.txt  0.51912993

Buscando climatico: Encontrados 0 hits.

Buscando climático: Encontrados 3 hits.
1. ./ficheros/cuatro.txt        0.6554797
2. ./ficheros/tres.txt  0.51912993
3. ./ficheros/dos.txt   0.41299203

Buscando por: Encontrados 0 hits.

Buscando aeropuerto: Encontrados 1 hits.
1. ./ficheros/uno.txt   2.3763716
```

Como se puede apreciar ahora para ṕorïo ha encontrado ningún resultado, esto es debido a que al estar esa palabra en nuestro fichero de stopWords la palabra se ignora por lo que no se busca en el resto de ficheros.

- What occurs if we put contaminacion or cambio climatico (without accents) in the search?

  Como podemos apreciar en los resultados anteriores, la palabra contaminación sin tilde la está ignorando, seguramente debido a que no están puestos los filtros de idioma (lo haremos en el siguiente punto para ver el resultado). De todas formas, podemos ver que cambio climático, sin la tilde, tuvo match las mismas veces que con tilde. Esto es debido a que, como es una palabra formada por dos, el analizador primero busca que haya un match con cambio, que le dará los cuatro hits, y luego con climático, que le arrojará un cero.

- What happens if we perform this search using the SpanishAnalyzer? Why does this occur?

```
Buscando Contaminación: Encontrados 1 hits.
1. ./ficheros/uno.txt   1.6493142

Buscando Contaminacion: Encontrados 1 hits.
1. ./ficheros/uno.txt   1.6493142

Buscando cambio climatico: Encontrados 3 hits.
1. ./ficheros/cuatro.txt        1.3262624
2. ./ficheros/dos.txt   1.0767463
3. ./ficheros/tres.txt  1.0317104

Buscando cambio climático: Encontrados 3 hits.
1. ./ficheros/cuatro.txt        1.3262624
2. ./ficheros/dos.txt   1.0767463
3. ./ficheros/tres.txt  1.0317104

Buscando cambio: Encontrados 3 hits.
1. ./ficheros/cuatro.txt        0.6631312
2. ./ficheros/dos.txt   0.5383732
3. ./ficheros/tres.txt  0.5158552

Buscando climatico: Encontrados 3 hits.
1. ./ficheros/cuatro.txt        0.6631312
2. ./ficheros/dos.txt   0.5383732
3. ./ficheros/tres.txt  0.5158552

Buscando climático: Encontrados 3 hits.
1. ./ficheros/cuatro.txt        0.6631312
2. ./ficheros/dos.txt   0.5383732
3. ./ficheros/tres.txt  0.5158552

Buscando por: Encontrados 0 hits.

Buscando aeropuerto: Encontrados 1 hits.
1. ./ficheros/uno.txt   2.4057715
```

Por fin, con este analizador, se observa que las palabras sin acentos también son encontradas y presentan el mismo score que sus equivalentes con acento. La stopword por fue eliminada, del mismo modo que las restantes. Sin embargo, los scores obtenidos son más altos que los del StandardAnalyzer (y del Simple). Esto se debe a que el SpanishAnalyzer está específicamente diseñado para el español: aplica stemming, normaliza acentos y utiliza una lista de stopwords optimizada, lo que reduce el ruido y permite que más palabras se unifiquen bajo la misma raíz. Como consecuencia, la frecuencia efectiva de los términos aumenta y, por tanto, los scores resultan más elevados.

- What happens if we re-index all files each time we run the program, instead of simply reopening the previously created index?

Esta cuestión ya fue abordada al inicio, y se indicó cómo se solucionó con el fin de obtener resultados más realistas y sin información repetida. En resumen, lo que ocurre es que los archivos se vuelven a añadir al índice, lo que provoca que se busque en el mismo archivo tantas veces como este haya sido indexado.

## 2.2. Conclusiones Clave del Análisis de Lucene

El análisis de las tres configuraciones de `Analyzer` (Simple, Standard y Spanish) sobre los mismos datos reveló la importancia crítica de la correcta gestión del procesamiento de texto para el idioma español.

### 2.2.1. Comparación de `Analyzers` y Procesamiento de Texto

Del análisis realizado se observa que tanto `SimpleAnalyzer` como `StandardAnalyzer` tratan los acentos como caracteres distintos, por lo que no recuperan resultados cuando se busca una palabra sin tilde. Este comportamiento contrasta con el de `SpanishAnalyzer`, que funciona adecuadamente con o sin tilde.

Además, `SpanishAnalyzer` incorpora una lista de stopwords propia del español, donde se incluye la palabra "por", de modo que dicha consulta no devuelve coincidencias. Este filtrado coincide con el del `StandardAnalyzer`, aunque el tratamiento específico para el español ofrece resultados más consistentes en nuestro caso. Por ello, se considera que el analizador más adecuado para estos textos es el `SpanishAnalyzer`.

# 3. Nuestra aplicación

Para probar el funcionamiento de la aplicación desarrollada, se utilizó el analizador `SpanishAnalyzer` y se procedió a testear las tres funcionalidades principales del menú:

1. **Indexación Inicial (Opción 1):** Se indexó el directorio `./doc` utilizando la opción 1. Al usar el modo `OpenMode.CREATE`, el índice se creaba sin duplicados.
2. **Añadir Documento (Opción 2 - APPEND):** Se añadió el archivo `./ficheros/uno.txt` utilizando la opción 2, con el modo `OpenMode.APPEND`.
3. **Búsqueda (Opción 3):** Se buscaron varios terminos para comprobar el funcionamiento de la aplicación, el primero fue çontaminacion".

```
====== MENÚ ======
1. Indexar directorio
2. Añadir documento al índice
3. Buscar término
4. Salir

Opción: 1

Ruta del directorio: ./doc

Indexación completada.
```

Opción 1: Menú principal

```
===== MENÚ =====
1. Indexar directorio ./doc
2. Añadir documento al índice
3. Buscar término
4. Salir

Opción: 2

Ruta del documento a añadir: ./ficheros/uno.txt

Documento añadido al índice.
```

Opción 2: Añadir documento

```
Término a buscar: contaminación

Buscando "contaminación": Encontrados 16 hits.
1. .\doc\52.txt Score: 4.1399336
Frecuencia: 3

2. .\doc\5.txt  Score: 3.8761427
Frecuencia: 10

3. ./ficheros/uno.txt   Score: 3.8424563
Frecuencia: 2

4. .\doc\11.txt Score: 3.6362722
Frecuencia: 3

5. .\doc\20.txt Score: 3.0566025
Frecuencia: 1
```

Figura 3.0.1: Búsqueda del término “contaminación”

Observamos que 16 archivos contienen la palabra con distintos scores y frecuencias. También se aprecia que el score asociado al fichero uno.txt difiere del obtenido anteriormente con el SpanishAnalyzer. Esto es completamente normal, ya que el valor depende del corpus que se esté indexando en cada caso. El cálculo de relevancia se basa en estadísticas globales como la frecuencia del término en cada documento, el número de documentos del corpus en los que aparece o la propia longitud de los textos indexados.

Por último, se realizaron búsquedas con otras palabras relevantes dentro del conjunto de documentos de la EINA, como “escuela” y “españa”.

```
Término a buscar: escuela

Buscando "escuela": Encontrados 21 hits.
1. .\doc\47.txt Score: 3.5022883
Frecuencia: 3

2. .\doc\36.txt Score: 3.4456031
Frecuencia: 3

3. .\doc\112.txt        Score: 3.1444945
Frecuencia: 2

4. .\doc\127.txt        Score: 3.0075095
Frecuencia: 3

5. .\doc\54.txt Score: 2.8690739
Frecuencia: 1

6. .\doc\115.txt        Score: 2.6794882
Frecuencia: 1

7. .\doc\34.txt Score: 2.6794882
Frecuencia: 1

8. .\doc\120.txt        Score: 2.4913447
Frecuencia: 1

9. .\doc\99.txt Score: 2.4625003
Frecuencia: 2

10. .\doc\78.txt        Score: 2.3483946
Frecuencia: 5
```

Resultados para “escuela”

```
Buscando "españa ": Encontrados 73 hits.
1. .\doc\135.txt        Score: 1.3520242
Frecuencia: 44

2. .\doc\133.txt        Score: 1.3493258
Frecuencia: 35

3. .\doc\84.txt Score: 1.3440038
Frecuencia: 6

4. .\doc\92.txt Score: 1.3424746
Frecuencia: 6

5. .\doc\89.txt Score: 1.3399299
Frecuencia: 12

6. .\doc\85.txt Score: 1.3232343
Frecuencia: 9

7. .\doc\97.txt Score: 1.3132819
Frecuencia: 20

8. .\doc\18.txt Score: 1.3101724
Frecuencia: 12

9. .\doc\99.txt Score: 1.3063037
Frecuencia: 14
```

Resultados para “españa”

Esos resultados parecen coherentes. Se observa que la palabra “españa” aparece en un número mucho mayor de archivos, superando la mitad del corpus. En cambio, “escuela” es menos frecuente, lo que explica que sus scores sean más altos: al aparecer en menos documentos, su peso relativo dentro del índice es mayor.

Se comprobó también el comportamiento con la palabra “organización”, y se verificó que, aun sin acento, es posible recuperar correctamente los documentos que la contienen.

```
Buscando "organizacion": Encontrados 58 hits.
1. .\doc\110.txt        Score: 1.7233087
Frecuencia: 3

2. .\doc\40.txt Score: 1.638256
Frecuencia: 3

3. .\doc\76.txt Score: 1.6263534
Frecuencia: 9

4. .\doc\89.txt Score: 1.60094
Frecuencia: 4

5. .\doc\134.txt        Score: 1.5628251
Frecuencia: 15
```

Figura 3.0.2: Resultados de la búsqueda para “organizacion”

Finalmente, se probó con una búsqueda de dos palabras ”muchas cosas”:

```
Término a buscar: "muchas cosas"

Buscando ""muchas cosas"": Encontrados 2 hits.
1. .\doc\73.txt Score: 5.3863397
2. .\doc\54.txt Score: 4.751228

Escribe 'menu' para volver al menú principal.
```

Figura 3.0.3: Resultados de la búsqueda para “muchas cosas”

Para buscar palabras juntas y no como términos independientes, es necesario introducirlas entre comillas dobles. En este caso se encontraron dos ocurrencias, lo que explica que los scores obtenidos sean muy altos; no obstante, en esta consulta no se calculó la frecuencia.

Además, se añadió una funcionalidad que permite seguir realizando búsquedas de forma continua hasta que el usuario escriba “menu”. Esta palabra no aparece en ningún archivo del corpus.

# 4. Data Mining with Weka

## 4.1. Clasificación Clásica: Flores Iris

Esta prueba establece la capacidad predictiva del algoritmo J48 frente a una línea base ZeroR, este ha sido un ejemplo guiado que se encontraba en el guion, el cual nos ha servido para familiarizarnos con la herramienta weka, puesto que ningún miembro del equipo había trabajado anteriormente con ella.

**Conclusión:** El resultado ha sido el esperado, al estar tan guiado era dificil confundirse, también llegamos a la conclusión de que los atributos de medidas físicas (longitud y anchura del sépalo y el pétalo) son variables altamente discriminativas y predictivas de la especie de la flor. El árbol de decisión `J48` infiere reglas muy precisas para la clasificación.

## 4.2. Reglas de Asociación y Clasificación del Clima

Esta sección analiza las condiciones climáticas que influyen en la decisión de jugar al tenis, utilizando el conjunto de reglas Apriori, también es un ejemplo muy guiado en el qué siguiendo paso a paso lo que se comenta en el guión se llega al resultado.

### 4.2.1. Análisis de Clasificación mediante J48

Al aplicar el algoritmo `J48` al conjunto de datos `weather.nominal.arff`, el árbol de decisión que nos queda, prioriza el Pronóstico (`outlook`) como variable principal, seguido

por la Humedad y el Viento.

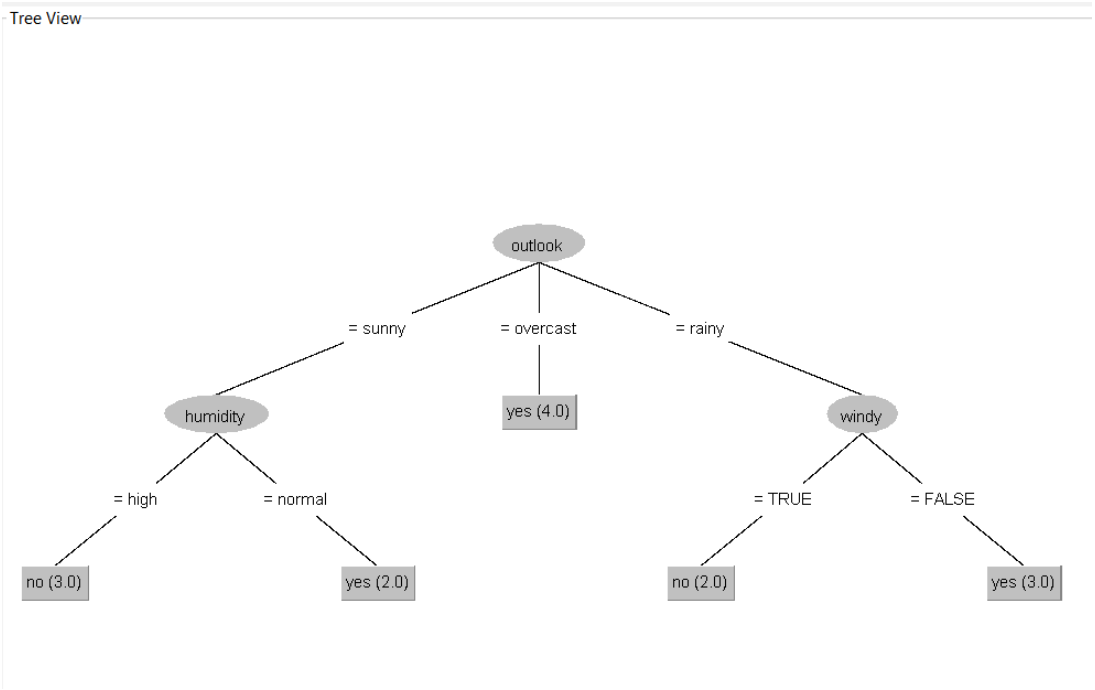



Figura 4.2.1: Arbol apriori

### 4.2.2. Análisis de Reglas de Asociación mediante Apriori

El algoritmo `Apriori` se ejecutó con las reglas que ya estaban predefinidas. Y como se puede ver en el árbol las principales reglas son:

- Si el pronóstico es **nublado** se juega siempre.
- Si el pronóstico es **soleado**:
  - Con **humedad alta** → no se juega.
  - Con **humedad normal** → se juega.
- Si el pronóstico es **lluvioso**:
  - Si hay **viento** (`windy = TRUE`) → no se juega.
  - Si **no hay viento** (`windy = FALSE`) → se juega.

Las reglas en su totalidad que hemos obtenido mediante las reglas apriori han sido:

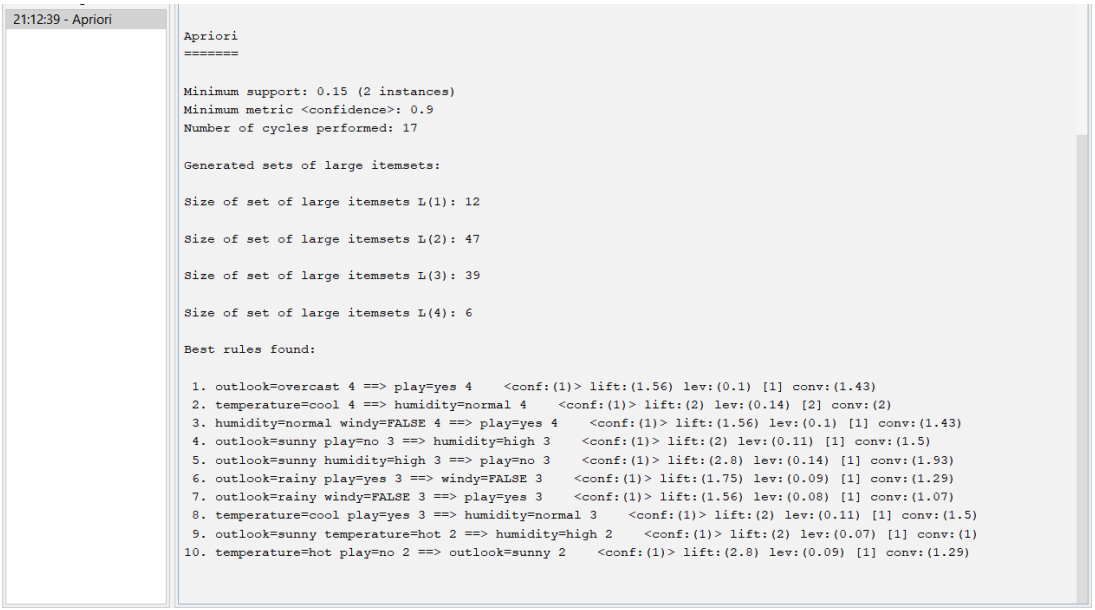


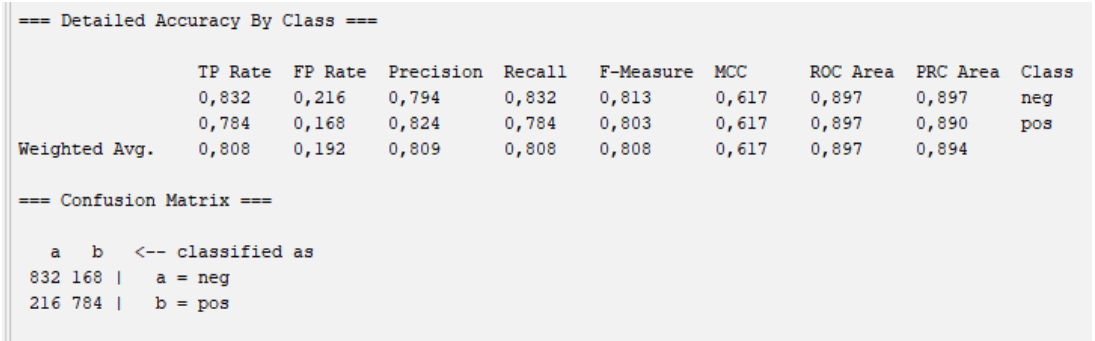

Figura 4.2.2: Reglas apriori

## 4.3. Sentiment Analysis and Natural Language Processing

Para este apartado hemos usado los datos de reseñas de películas que nos han proporcionado tanto los negativos como los positivos.

Para poder introducirlos en weka se han seguido las pautas del guión que están muy bien explicadas, ya que weka no acepta directorios porlo que teniamos que convertir el directorio con las dos carpetas neg y pos en un archivo .arff usando la opción de TextDirectoryLoader y posteriormente StringToWordVector para convertir los datos.

Después se aplico el algoritmo de clasificación de Naive Bayes en el atributo @@class@@ que nos indicaba si la opinión de una película era positiva o negativa y los resultados han sido:



```
=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                0,832    0,216    0,794      0,832    0,813      0,617    0,897     0,897     neg
                0,784    0,168    0,824      0,784    0,803      0,617    0,897     0,890     pos
Weighted Avg.   0,808    0,192    0,809      0,808    0,808      0,617    0,897     0,894

=== Confusion Matrix ===

   a   b   <-- classified as
 832 168 |   a = neg
 216 784 |   b = pos
```

Figura 4.3.1: Naive Bayes

Después de esto hemos repetido la misma operación pero removiendo palabras irrelevantes del vector. Esto lo hemos hecho en la configuración de `StringToWordVector`

añadiendo en el delimitador estos símbolos y números: `\r\n\t.,;:':"()?!@&-+#$%_=^|~<>0123456789`, lo cuál ha empeorado bastante; las estadísticas han caído un 20 %.

```
=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                0,679    0,455    0,599      0,679    0,636      0,226    0,654     0,641     neg
                0,545    0,321    0,629      0,545    0,584      0,226    0,654     0,657     pos
Weighted Avg.   0,612    0,388    0,614      0,612    0,610      0,226    0,654     0,649

=== Confusion Matrix ===
```

Figura 4.3.2: Naive Bayes

Para la optimización del análisis de lenguaje natural, se experimentó con los parámetros del filtro `StringToWordVector` con el fin de mejorar la precisión y el recall globales. Se encontró que la precisión global mejoró significativamente al modificar el parámetro `wordsToKeep` de su valor inicial (1000) a 2000. Esta mejora se debe a que el modelo permite que el clasificador `NaiveBayes` incorpore 1000 *tokens* adicionales.

```
=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                0,755    0,130    0,853      0,755    0,801      0,629    0,896     0,896     neg
                0,870    0,245    0,780      0,870    0,823      0,629    0,896     0,893     pos
Weighted Avg.   0,813    0,188    0,817      0,813    0,812      0,629    0,896     0,895

=== Confusion Matrix ===
```

Figura 4.3.3: Naive Bayes

# 5. Delving Deeper into Classification Trees

El objetivo de este ultimo apartado es analizar de forma más detallada los algoritmos de clasificación. Para ello debemos intentar adivinar el tipo de fármaco que se le debe dar a un paciente mediante distintos parámetos como la edad, sexo, colesterol...

Se han ejecutado los primeros apartados tal y como indicaba el guión para ir viendo y entendiendo los datos que nos arrojaban como que con ZeroR se conseguía un 45,5 % de recomendación para usar el fármaco y, y que si eso no funcionaba pues pasaria al siguiente de la lista que es el X. Después también se probo con J48 tal como decía en el apartado 3.

Posteriormente, fuimos a la pestaña de visualizacion para ver de manera gráfica si había alguna variable que tuvieran una alta correlación entre sí.

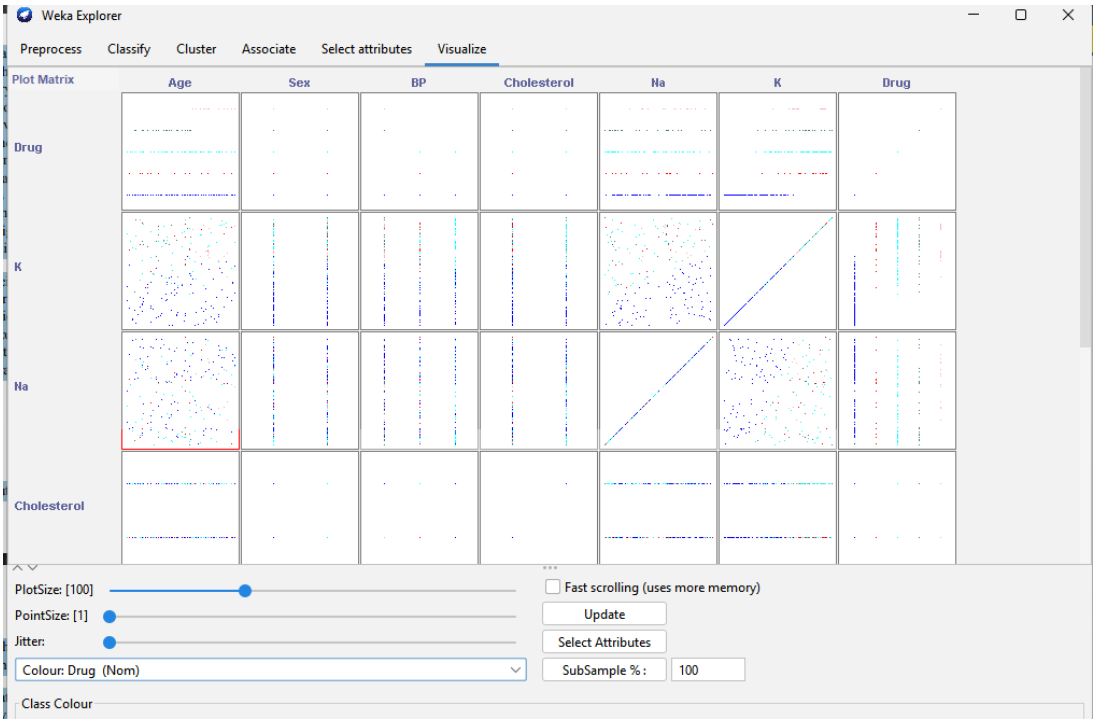



Figura 5.0.1: Gráfica

Observando la gráfica, se puede apreciar la relación que hay entre los niveles de potasio y sodio ya que forman una línea completamente recta, por lo que podremos simplificarlas sustituyéndolas por el cociente entre las dos.

Para simplificar el modelo y aumentar su precisión, en la ventana Preprocess, se utilizó el filtro AddExpression para crear un nuevo atributo (`Na_to_K`) a partir de la división de los niveles de Potasio (K) y Sodio (Na). Esta nueva variable es un indicador más claro y potente que las dos variables por separado, permitiendo que el clasificador `J48` genere reglas de decisión más sencillas y fiables. Y podemos ver que funciona viendo que las estadísticas han mejorado a un 99 % y el árbol generado es bastante más sencillo:

```
=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,989    0,000    1,000      0,989    0,994      0,990    0,995     0,994     drugY
                 1,000    0,005    0,941      1,000    0,970      0,968    0,997     0,941     drugC
                 1,000    0,000    1,000      1,000    1,000      1,000    1,000     1,000     drugX
                 1,000    0,006    0,958      1,000    0,979      0,976    0,997     0,958     drugA
                 0,938    0,000    1,000      0,938    0,968      0,966    0,969     0,943     drugB
Weighted Avg.    0,990    0,001    0,991      0,990    0,990      0,987    0,994     0,983

=== Confusion Matrix ===

  a  b  c  d  e   <-- classified as
 90  1  0  0  0 |  a = drugY
  0 16  0  0  0 |  b = drugC
  0  0 54  0  0 |  c = drugX
  0  0  0 23  0 |  d = drugA
  0  0  0  1 15 |  e = drugB
```

Figura 5.0.2: Estadísticas J48

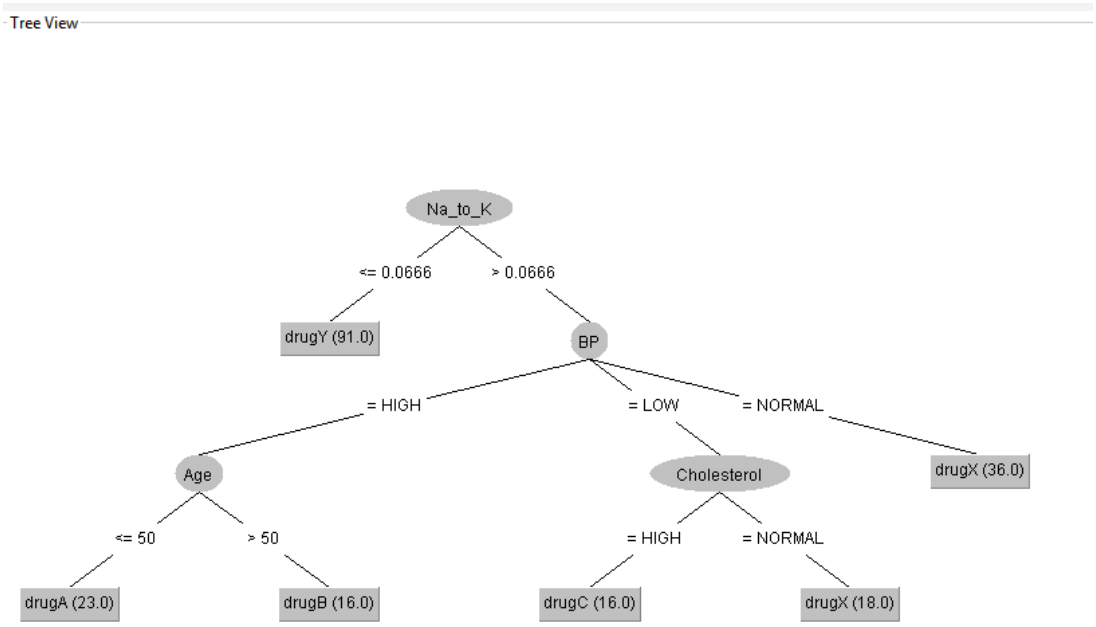



Figura 5.0.3: Árbol simplificado

Podemos apreciar ahora en el nuevo árbol que si nuestra nueva variable introducida Na_to_K es menor o igual a 0.0666 se recomienda el fármaco Y al 91 % y si es mayor ya se fijará en otras cosas como BP, colesterol y edad, este nuevo modelo simplifica bastante el anterior que como se puede observar en la siguiente imagen era bastante más complejo:

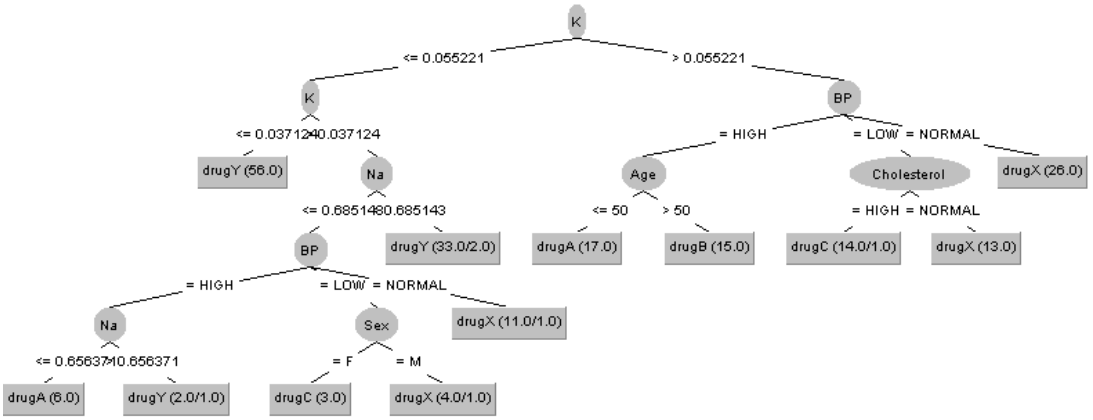



Figura 5.0.4: Árbol inicial

# 6. Metodología de trabajo

## 6.1. Herramientas de trabajo

Para la realización de las prácticas 6 y 7 se empleó el entorno Eclipse para el desarrollo del bloque correspondiente a Lucene y la herramienta Weka para el bloque de análisis de

datos. La memoria se elaboró posteriormente utilizando la plataforma Overleaf.

## 6.2. Distribución del trabajo

En cuanto a la distribución del trabajo, la práctica 6 fue desarrollada principalmente por Chloé, mientras que José revisó y matizó algunos aspectos del código y elaboró la memoria final. La práctica 7 fue realizada conjuntamente por José y Sergio.

La mayor parte del trabajo se llevó a cabo durante las sesiones de laboratorio, destinándose aproximadamente dos horas adicionales fuera del horario presencial para completar y depurar el contenido de la memoria.

Cuadro 1: Horas Estimadas de Trabajo

| **Miembro** | **Horas Estimadas** |
|---|---|
| Sergio Saura | 6 |
| José Secadura | 6 |
| Chloé Bolle-Reddat | 6 |
| **Total** | **18** |