Universidad de Murcia - Facultad de Informática

DESARROLLO DE SISTEMAS INTELIGENTES

Alertas de ECG: Informe 3

Curso académico 2025-26

Profesores: Ricardo Javier Sendra Lázaro – Aurora González Vidal

Juan Rondán Ramos (j.rondanramos@um.es)

Domingo Pujante López de Ochoa (domingo.pujantel@um.es)

David Madrid del Amor (d.madridamor@um.es)

Contenido

1. INTRODUCCION

Este proyecto se centra en el desarrollo de un Sistema Inteligente para la detección automática de patrones de riesgo en señales de electrocardiograma (EGC). El núcleo del sistema reside en la implementación de un motor de reglas que, mediante una serie de reglas establecidas, es capaz de ofrecer diagnósticos a partir de los datos extraídos del ECG.

El sistema está diseñado para interpretar automáticamente las características esenciales de la señal cardiaca, identificando posibles anomalías que puedan representar riesgos para la salud. A través de un proceso de análisis estructurado, se evalúan los diferentes componentes del ECG y sus relaciones, permitiendo reconocer patrones asociados a diversas condiciones médicas.

2. LECTURA DE LOS DATOS DE ENTRADA

Además de los estudios previos teóricos, el primer paso de este proyecto ha consistido en desarrollar un mecanismo capaz de leer la entrada, omitiendo líneas intrascendentes (líneas en blanco y las que comienzan por '#'), detectar posibles errores en el fichero, y en su defecto, extraer sus datos.

2.1- Extracción de datos

La extracción de datos se ha llevado a cabo mediante la clase Java **LectorECG**, donde, con la ayuda de expresiones regulares, se detectan las líneas del fichero de entrada que ofrecen los atributos de una Onda de forma correcta. Una vez verificado, se crea un objeto Onda por línea correcta con sus respectivos atributos y se almacenan en una lista. El método encargado de realizar este proceso, devuelve una instancia de una nueva clase, **EntradaElectro**, cuyo único atributo es la lista donde se han almacenado las ondas. Esta otra clase únicamente cuenta con los métodos get y set habituales.

2.2- Creación de "KieSession"

La configuración e inicialización del Sistema Inteligente que combina Java con el motor de reglas Drools, se realiza directamente sobre el main del proyecto.

En primer lugar, se establece la conexión con el motor de reglas Drools. A través de **KieServices**, se inicia el servicio de Drools, se carga la configuración del proyecto desde **kmodule.xml** con **kContainer** y se crea una sesión de trabajo **kSession** de nombre *"***ksession-rules"** donde se ejecutarán todas las reglas.

2.3- Inserción de hechos

Previamente a la inserción de hechos en la base de hechos, se ha creado un objeto de cada clase necesaria para el análisis ECG.

Para las clases principales, se ha instanciado un objeto individual de cada una y se ha insertado directamente en la sesión de Drools, incluyendo **resultado, intervalo, entrada y contador de ciclos**. En el caso particular de las **ondas**, al estar almacenadas en la lista de **entrada** (instancia de **EntradaElectro***)*, se ha realizado un bucle que recorre dicha lista, y, para cada onda contenida en ella, ejecuta una operación de inserción. De esta manera, cada onda se convierte en un hecho independiente que el motor de reglas puede evaluar por separado.

2.4- Lanzamiento de reglas y cierre de sesión

Con la ejecución de **fireAllRules(),** se dispara el motor de interferencia de Drools, que evalúa y ejecuta todas las reglas aplicables sobre los hechos previamente insertados. Una vez completado el proceso de razonamiento, **dispose()** cierra la sesión de manera controlada, liberando los recursos utilizados durante la ejecución.

3. REGLAS DE MEDIDAS

3.1- Asignar ciclo en orden

La regla **Asignar Ciclo en orden** se encarga de buscar una onda que no tenga ciclo, comprobando que sea la más antigua sin que se le haya asignado un ciclo, para asignárselo correspondientemente. Está regla asignará el número de ciclo a todas las ondas, con excepción de las tipo P, cuya asignación se llevará a cabo en la siguiente regla **Si es onda P aumentar los ciclos.**

3.2- Si es onda P aumentar los ciclos

La regla **Si es onda P aumentar los ciclos**, tiene como propósito buscar ondas P a las que no se le haya asignado aún un ciclo. Al ser este tipo de onda la que da comienzo a un nuevo ciclo, se comprobará que sea la onda de tipo P más antigua sin ciclo asignado, para aumentar el contador global de ciclos, asignando el nuevo valor. La unión de esta regla con la anterior, hacen posible una correspondiente asignación de ciclos a cada una de las ondas presentes.

3.3- Calcular Número de Ciclos

La regla **Calcular numCiclos** busca un objeto resultado cuya variable **numCiclos** sea 0. Además, recopila todas las ondas que existen y las almacena en una lista llamada *ondas*. A partir de esta lista de ondas, se toma la longitud de la lista, que se dividirá entre el número de ondas que componen un ciclo (**5 – PQRST**). Si existiera parte decimal en este valor, se realizará un redondeo al valor superior, obteniendo el número total de ciclos. Este valor será asignado al atributo **numCiclos** del objeto **Resultado** previamente encontrado.

3.4- Calcular Ritmo Cardiaco

La regla **Calcular Ritmo Cardiaco** busca un objeto **Resultado** cuyo **numCiclos** sea mayor que 0 y su **ritmoCardiaco igual a 0.** Se recopilan todas las ondas que existen y se almacenan en una lista **ondas**. Por precaución y para evitar una lista de ondas desordenadas, se introduce un sort, que ordena las ondas en función del milisegundo de inicio. También, para evitar ciclos incompletos, se identifican las ondas que no forman un ciclo completo y se eliminan, sin tenerlas en cuenta. Una vez hecho esto, se inicia con el cálculo, restando el final de la última onda con el inicio de la primera, obteniendo una **duración total**. A continuación, se divide este valor por el número de ciclos del resultado encontrado, obteniendo una **duración media.** Finalmente se divide el número 60.000 (proveniente de cambios de unidades) entra la **duración media** calculada. Este valor se asigna al atributo **ritmoCardiaco** del **Resultado** encontrado al inicio.

4. REGLAS DE COMPLEJOS-INTERVALOS

4.1- Intervalo QT

Para interceptar un intervalo QT se buscan dos ondas, una onda Q y otra onda T pertenecientes al mismo ciclo. Además, se comprueba que no existe ningún objeto **Intervalo** de tipo “QT” con el número de ciclo de las ondas interceptadas. Una vez comprobadas dichas condiciones, se calcula la duración del intervalo, restando el final de la Onda T con el inicio de la Onda Q, creando una nueva instancia de **Intervalo** de tipo “QT” con la duración calculada.

4.2- Intervalo ST

De manera similar al intervalo QT, para interceptar un intervalo ST, se buscan dos ondas, una S y una T, pertenecientes al mismo ciclo. Adicionalmente, también se comprueba que no existe un objeto **Intervalo** de tipo “ST” con el número de ciclo de las ondas interceptadas. Una vez cumplidas las condiciones, se calcula la duración del intervalo, restando el final de la onda T con el inicio de la onda S, creando una nueva instancia de **Intervalo** de tipo “ST” con la duración calculada.

4.3- Complejo QRS

Para interceptar un complejo QRS, se buscan dos ondas, una Q y una S, que pertenezcan al mismo ciclo, comprobando, además, que no existe ningún objeto **Intervalo** del tipo “QRS” con el mismo número de ciclo que las ondas interceptadas. Una vez cumplidas las condiciones, se calcula la duración del complejo restando el final de la onda S con el inicio de la onda Q, creando una nueva instancia de **Intervalo** de tipo “QRS” con la duración calculada.

4.4- Intervalo RR

Para interceptar un intervalo RR es necesario buscar dos ondas tipo R, cuyo número de ciclo sea contiguo. Al igual que en casos anteriores, adicionalmente se comprueba que no exista una instancia de **Intervalo** del tipo RR que incluya dos ondas tipo R con esos dos ciclos concretos. Una vez verificado, se calcula la duración del intervalo restando el final de la onda R con mayor número de ciclo con el inicio de la onda R con menos número de ciclo, creando una nueva instancia de **Intervalo** de tipo RR con la duración calculada.

4.5- Promedio RR

La regla **Calcular promedio RR** recopila todos los intervalos de tipo RR que existen, almacenándolos en una lista de nombre **intervalosRR**. Adicionalmente se comprueba que no exista un **Intervalo** del tipo “Promedio_RR”. Para realizar el cálculo, se suman todas las duraciones de todos los intervalos de tipo RR, dividiendo el valor obtenido entre el tamaño de la lista. Finalmente se crea una instancia de **Intervalo** del tipo “Promedio_RR”, con la duración promedio calculada.

5. REGLAS PARA LA DETECCIÓN DE PATOLOGÍAS

5.1- Bradicardia

La regla **Detectar Bradicardia** identifica automáticamente posibles casos de bradicardia al detectar un ritmo cardiaco demasiado bajo, garantizando que la enfermedad se registre solo una vez. Esta regla se activa al identificar un objeto **Resultado** con un **ritmoCardiaco** menor a 60 y cuando la enfermedad no está contenida en su lista **enfermedades**. Si se cumple la condición, la regla modifica el objeto **Resultado** agregando Bradicardia a la lista de **enfermedades**.

5.2- Taquicardia

La regla **Detectar Taquicardia** identifica automáticamente posibles casos de taquicardia al detectar un ritmo cardiaco elevado, garantizando que la enfermedad se registre solo una vez. Esta regla se activa al identificar un objeto **Resultado** con un **ritmoCardiaco** mayor a 100 y cuando la enfermedad no está contenida en su lista **enfermedades**. Si se cumple la condición, la regla modifica el objeto **Resultado** agregando Taquicardia a la lista de **enfermedades.**

5.3- Hipocalcemia

La regla **Detectar Hipocalcemia** identifica automáticamente posibles casos de hipocalcemia basándose en una condición principal y clave: que la duración del **intervalo QT** sea mayor a 450ms. Esta regla se activa al encontrar una instancia de **Intervalo** del tipo “QT” cuya **duración** sea mayor a 450ms. A esto se le suma otras condiciones lógicas como tener un objeto **Resultado** con un **ritmoCardiaco** positivo o que la enfermedad no haya sido detectada previamente y no se encuentre contenida en su lista de **enfermedades**. Si se cumple la condición, la regla modifica el objeto **Resultado** agregando Hipocalcemia a la lista de **enfermedades.**

5.4- Hipopotasemia

La regla **Detectar Hipopotasemia** identifica automáticamente posibles casos de hipopotasemia basándose en 3 condiciones principales: un **valor** relativamente alto y **negativo** de la **onda T**, un **intervalo ST descendente** y un valor **pico** de la **onda T** mayor que el de la **onda R** (en valor absoluto). Esta regla se activa al encontrar una **Onda T, Onda R y onda S** pertenecientes al mismo ciclo donde, el **pico** de la **Onda T** sea menor a -10, el **pico** de la **Onda T** sea menor al **pico** de la **Onda S** y la división entre el **pico** de la **Onda T** (en valor absoluto) y el **pico** de la **Onda R** sea mayor a 1. A todo esto se le suma otras comprobaciones base al requerir de una instancia de **Resultado** cuyo **ritmoCardiaco** sea positivo o que la enfermedad no esté ya contenida en la lista de **enfermedades**. Si se cumplen las tres condiciones al mismo tiempo, la regla modifica el objeto **Resultado** agregando Hipopotasemia a la lista de **enfermedades**.

5.5- Contracción Ventricular Prematura

La regla **Detectar Contracción Ventricular Prematura** identifica automáticamente posibles casos de CVP basándose en 2 condiciones principales: La primera condición tenía que ver con la prematuridad del promedio de los **intervalos RR**, comprobando si existía un **ciclo** RR menor a 0.8 multiplicado por el **promedio RR**. Sin embargo, está condición resultó ser muy poco restrictiva ya que se cumplía en demasiados casos, por lo que se tuvo que establecer un rango concreto, entre 600-700, observando este intervalo como zona de riesgo de CVP. La segunda

condición consistía en que el **intervalo QRS** debía tener una duración mayor a 120ms, sin embargo, esto no ocurre en ningún caso de CVP. El debug realizado indicaba que, para diagnósticos de CVP, este valor rondaba los 50-70ms, por lo que, al observar que este patrón únicamente se repetía en esta enfermedad, se ha tenido que adaptar la condición estableciendo un rango menor o igual a 70ms para poder diagnosticar esta enfermedad concreta.

Por tanto, esta regla se activará cuando se encuentre un **Intervalo RR** cuya **duración** este entre 600-700ms y cuando a la vez se encuentre un **intervalo QRS** cuya **duración** sea menor o igual a 70ms. A esto se le añade la validación habitual de encontrar una instancia de **Resultado** con un **ritmoCardiaco** positivo o que la enfermedad no este incluida en su lista de **enfermedades.**

5.6- Isquemia Coronaria

La regla **Detectar Isquemia Coronaria** identifica automáticamente posibles casos de isquemia basándose en 2 condiciones principales: La primera condición es que exista una diferencia concreta entre los valores pico de la **onda T** y la **onda S**. En un principio se consideraba que bastaba con que esta diferencia fuera menor o igual a -0.1, sin embargo, resultaba insuficiente, por lo que se ha tenido que ampliar hasta menor o igual a -5. La segunda condición tiene que ver con el **pico de la onda T**, donde en un principio únicamente se requería que fuera negativa, sin embargo, al generar conflictos con otras enfermedades (concretamente con hipopotasemia), se ha tenido que modificar a que dicho pico sea menor a -10.

Por tanto, esta regla se activará cuando se encuentra una **onda S** y una **onda T** pertenecientes al mismo **ciclo**, donde la diferencia entre el **pico** de la **onda T** y el **pico** de la **onda S** sea menor o igual que -5 y el **pico** de la **onda T** sea menor a -10. A esto se le añade la validación habitual de encontrar una instancia de **Resultado** con un **ritmoCardiaco** positivo o que la enfermedad no este incluida en su lista de **enfermedades.**

5.7- Infarto Agudo de Miocardio

La regla **Detectar Infarto Agudo de Miocardio** identifica automáticamente posibles casos de infarto basándose en cualquiera de dos condiciones: La primera condición es que exista una combinación específica en las **ondas T, S, R y Q** donde la suma de los **picos de T y S** sea mayor o igual a 0.2 y el pico de R sea mayor que 1. Inicialmente, el criterio para la suma de T y S era más estricto (mayor a 0.5) requiriendo que el **pico** de T fuera negativo, sin embargo, se eliminó este último, así como que se añadió el requisito de que el **pico** de R fuera mayor a 1 para aumentar la precisión. La segunda condición contempla la presencia de ondas Q patológicas, definidas por un **pico** menor a -0.1, una **duración** mayor o igual a 40ms y una relación entre el **pico** de Q (en valor absoluto) y el **pico** de R mayor o igual a 0.25.

Por tanto, esta regla se activará cuando se encuentre un conjunto de **ondas T, S, R y Q** pertenecientes al mismo **ciclo**, donde la suma de los **picos** de T y S sea mayor o igual a 0.2 y el **pico** de R mayor a 1; o bien donde se encuentre una onda Q con **pico** menor a -0.1 y una duración mayor o igual a 40ms y cuya relación (en valor absoluto) con el **pico** de la onda R del mismo **ciclo** sea mayor o igual a 0.25.

6. MANUAL DE USUARIO

Para probar el funcionamiento del Sistema Inteligente, se deberá hacer desde el propio Eclipse, previamente configurado correctamente con las versiones adecuadas de Drools indicadas en clases de prácticas.

Posteriormente se deberá importar el proyecto dentro de Eclipse, siendo imprescindible respetar la jerarquía de directorios, así como los nombres de los mismos. Adicionalmente y de forma extra, se facilita el enlace al repositorio de github donde se ha desarrollado el proyecto y donde se encuentra la versión finalizada. (Alertas-ECG).

Una vez el proyecto se haya importado correctamente, se deberá compilar y ejecutar.

Al hacer esto, la consola del mismo eclipse, solicitará al usuario que se introduzca el nombre de un fichero. Dicho fichero deberá estar contenido dentro de la carpeta **inputs**, que, si se ha importado correctamente, contará con todos los ficheros de prueba disponibles.

Únicamente será necesario introducir el nombre del fichero, es decir, no hay que introducir la ruta (inputs/nombre_fichero), no hay que añadir la extensión del fichero (nombre_fichero.txt) etc. Estrictamente solo se debe introducir el nombre del fichero, ejemplo: bradicardia. De lo contrario se lanzará una excepción **"Error procesando el fichero"**.

Al escribir correctamente el nombre y presionar la tecla "enter", se mostrará un mensaje de que el fichero se ha leído correctamente y que se está iniciando el diagnóstico. Aparecerán adicionalmente unos warnings de logging, que no afectan a la funcionalidad. Inmediatamente después se mostrarán los resultados diagnosticados (número de ciclos, ritmo cardiaco, posible enfermedad detectada, en qué ciclo (si se pudiera concretar), y el por qué de su detección). Finalmente aparecer un cuadro resumido con el resultado del análisis. Un ejemplo de salida, siguiendo estos pasos correctamente, es:

```
Introduce el nombre del fichero a evaluar:
bradicardia
Fichero inputs/bradicardia.txt leído correctamente. Ondas totales:79
Iniciando el diagnostico...
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Numero de ciclos calculado: 16.0
Ritmo Cardiaco calculado: 46.26506
Detectada posible bradicardia. Ritmo cardiaco bajo: 46.26506

===== RESULTADO DEL ANÁLISIS =====
Enfermedad: [BRADICARDIA]
Número de ciclos: 16.0
Ritmo Cardiaco: 46.26506
==================================
```

7. BIBLIOGRAFÍA

- Guion Proyecto
- Documentación de Drools
- Hipopotasemia
- Análisis ECG – Ondas y Segmentos