Trabajo Final Integrador

Análisis de Electrocardiograma

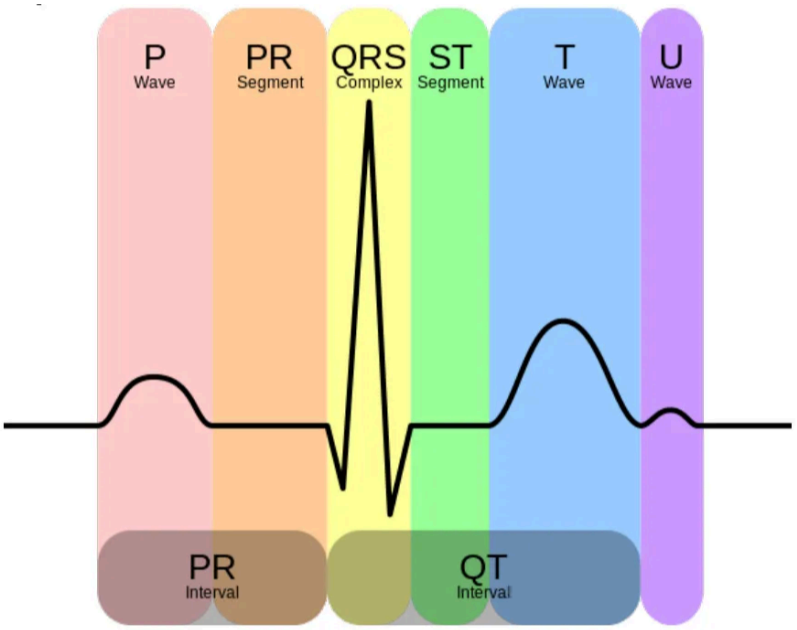
Grupo 8 - Felicitas, Angelina, Marisa, Melina, Alma

Fecha de Entrega: estimativo 10/06

- ¿Qué es un ECG?

Un electrocardiograma, también llamado ECG o EKG, de 12 derivaciones, es una prueba simple no invasiva que registra la actividad eléctrica del corazón.

https://www.physio-pedia.com/Electrocardiogram



Consignas

- Código genérico (que procese cualquier registro o estudio)

- A realizar en cualquier IDE

PDB 2025 - Enunciado TFI.pdf

Librería a utilizar obligatoriamente: *sleepecg*

https://sleepecg.readthedocs.io/en/stable/

N = normal

A = arritmia

Apartado 3

- Se calcula el promedio para sacar la línea base
- NO considerar el intervalo entre el pulso (sin matar parte de la señal)

La biblioteca `sleepecg` de Python está diseñada para trabajar con **señales de electrocardiograma (ECG)**, especialmente en el contexto del **análisis del sueño**. Es útil para investigadores y profesionales que trabajan con polisomnografía o estudios del sueño y desean extraer información del ECG.

🧠 ¿Qué hace `sleepecg`?

`sleepecg` permite:

- **Procesar registros de ECG**, típicamente extraídos de estudios de sueño (como los del PhysioNet).
- **Detectar latidos (R-peaks)** en una señal de ECG.
- **Calcular el intervalo RR** (el tiempo entre dos latidos consecutivos).
- **Interpolar series RR** para análisis espectral o del sistema nervioso autónomo.
- **Clasificar latidos anómalos** (ej. latidos ectópicos).
- Realizar **análisis de variabilidad de la frecuencia cardíaca (HRV)**.

🔧 Funciones principales de `sleepecg`

1. `sleepecg.detect_r_peaks(signal, sampling_rate)`

Detecta los picos R en la señal de ECG, que marcan los latidos del corazón.

2. `sleepecg.compute_rr_intervals(r_peaks)`

Calcula los intervalos RR a partir de los picos R.

3. `sleepecg.clean_rr_intervals(rr_intervals)`

Filtra o corrige artefactos y latidos anormales en los intervalos RR.

4. `sleepecg.interpolate_rr(rr_intervals, time)`

Interpola los intervalos RR a una frecuencia de muestreo regular para análisis más sofisticados.

5. `sleepecg.hrv_metrics(rr_intervals)`

Calcula medidas de variabilidad de la frecuencia cardíaca (HRV), como:

- SDNN (desviación estándar de los RR)
- RMSSD (raíz cuadrada de la media de las diferencias cuadradas)
- pNN50, entre otras.

6. `sleepecg.load_dataset()` ***(cuando trabaja con bases como PhysioNet)***

Facilita la carga de datos de ECG para estudios del sueño.

Web

https://www.kaggle.com/code/behl1anmol/mit-bih-arrhythmia-classification *

Lleva a un **notebook de Kaggle** titulado **"MIT-BIH Arrhythmia Classification"**. Este notebook se centra en un proyecto de **clasificación de arritmias cardíacas** utilizando el conocido dataset **MIT-BIH Arrhythmia Database**

🧠 Proyecto: Clasificación de arritmias cardíacas

📂 Dataset utilizado:

- **MIT-BIH Arrhythmia Dataset**: Un conjunto de datos ampliamente usado en medicina y machine learning, que contiene grabaciones de ECG (electrocardiogramas) anotadas manualmente, representando distintos tipos de latidos cardíacos normales y anómalos.

⚙️ Objetivo:

- Clasificar correctamente los tipos de latidos cardíacos (por ejemplo: normales, latidos prematuros ventriculares, etc.) a partir de señales de ECG.

🧪 **Pasos que se siguen en el notebook:**

1. **Importación de librerías** necesarias (NumPy, Pandas, Scikit-learn, TensorFlow, etc.).
2. **Carga y preprocesamiento de los datos**, incluyendo la lectura de señales de ECG y anotaciones.
3. **Segmentación** de la señal para centrarse en latidos individuales.
4. **Codificación de etiquetas** para representar los diferentes tipos de latidos.
5. **Construcción del modelo de deep learning**, generalmente una red neuronal convolucional (CNN).
6. **Entrenamiento y validación del modelo** sobre los datos.
7. **Evaluación del rendimiento**, mostrando métricas como accuracy, precision y recall.
8. **Visualización de resultados**, como curvas de aprendizaje o matrices de confusión.

📊 **Modelo usado:**

- Típicamente se implementa una **CNN (Red Neuronal Convolucional)**, adecuada para analizar datos secuenciales como señales ECG.

- **CNN**: *Red Neuronal Convolucional*

Es un tipo de red neuronal artificial especialmente diseñada para trabajar con **datos con estructura espacial o temporal**, como **imágenes** o **señales.**

Se crea un modelo que es como una Clase

Al crear un modelo agrega una red con filtros *(Building the Model Architecture*)*

Model.add( ) : agrega una línea a la red neuronal

Fitting the model: modificando parámetros de la red → resultado coherente (entrenar la red)

Estructura Base

📦 Estructura general del proyecto

🗂️ 1. Carga y organización de archivos

Objetivo: Leer archivos `.csv` (ECG) y `.txt` (anotaciones).

- `lector_archivos.py`
  - Función para abrir archivos `.csv` con pandas.
  - Función para leer anotaciones desde `.txt`.
  - Uso del módulo `os` para navegar carpetas.

🧬 2. Clase `Estudio`

Objetivo: Representar un estudio completo de ECG.

- `estudio.py`
  Clase `Estudio`:
  - Atributos:
    - Señal de ECG.
    - Anotaciones.
    - Frecuencia de muestreo.
  - Métodos:
    - `calcular_frecuencia_cardiaca(inicio=None, fin=None)`
    - `detectar_latidos_por_txt()`
    - `detectar_latidos_por_maximos()`
    - `detectar_latidos_sleepecg()`
    - `comparar_detecciones()`
    - `visualizar_senal_completa()`
    - `visualizar_segmento(tiempo_inicio, tiempo_fin)`

❤️ 3. Clase `Pulso`

Objetivo: Representar un latido individual.

- `pulso.py`
  Clase `Pulso`:
  - Atributos:
    - Tipo: normal / anormal.
    - Datos de la señal.
  - Métodos:
    - `graficar()`
    - `normalizar()`
    - `suavizar(ventana=5)`
      (Usando `pandas.Series.rolling().mean()`)

🧠 4. Inteligencia artificial (IA con CNN)

Objetivo: Ejecutar y comprender el Jupyter Notebook de Kaggle indicado: 📎 MIT-BIH Arrhythmia Classification - CNN

- `analisis_ia.ipynb`
  - Descargar y adaptar el dataset.
  - Entender cada paso del código:
    - Preprocesamiento.
    - Arquitectura CNN.
    - Entrenamiento y evaluación.
  - Agregar comentarios explicativos.
  - Mostrar resultados y visualizaciones.

🧪 5. Archivo principal para ejecución

Objetivo: Integrar todo el proyecto y facilitar pruebas.

- `main.py`
  - Carga del estudio.
  - Visualización de la señal.
  - Comparación de detección de pulsos.
  - Extracción y visualización de pulsos normales/anormales.
  - Ejecución de métodos de `sleepecg`.