MIT-BIH Arrhythmia Database

https://physionet.org/content/mitdb/1.0.0/

La base de datos de arritmias del MIT-BIH contiene 48 extractos de media hora de registros de ECG ambulatorios de dos canales, obtenidos de 47 sujetos estudiados por el Laboratorio de Arritmias de BIH entre 1975 y 1979. Se eligieron veintitrés registros al azar de un conjunto de 4000 registros de 24 horas de ECG ambulatorios recopilados de una población mixta de pacientes hospitalizados (alrededor del 60%) y ambulatorios (alrededor del 40%) en el Hospital Beth Israel de Boston; los 25 registros restantes se seleccionaron del mismo conjunto para incluir arritmias menos comunes pero clínicamente significativas que no estarían bien representadas en una muestra aleatoria pequeña.
Las grabaciones se digitalizaron a 360 muestras por segundo por canal con una resolución de 11 bits en un rango de 10 mV. Dos o más cardiólogos anotaron cada registro de forma independiente. Los desacuerdos se resolvieron para obtener anotaciones de referencia legibles por computadora para cada latido (aproximadamente 110.000 anotaciones en total) incluidas en la base de datos.

Para más información sobre las anotaciones y los registros, puedes acceder al siguiente enlace: https://archive.physionet.org/physiobank/database/html/mitdbdir/mitdbdir.htm

Actividad:
El Trabajo Final Integrador se realizará en Python, en Spyder, Visual o cualquier otro IDE que desees utilizar. Lo importante es que el código sea lo “suficientemente genérico” para que pueda procesar cualquier estudio/registro (par de archivos .csv y .txt) sin problemas.

Algunas bibliotecas de Python que podremos utilizar en esta sección son las siguientes:

- os: módulo que provee una manera versátil de usar funcionalidades dependientes del sistema operativo para leer carpetas y archivos.
    - https://docs.python.org/es/3.10/library/os.html
- csv: para lectura de archivos csv.
    - https://docs.python.org/es/3.8/library/csv.html
- matplotlib: módulo de visualización.
    - https://matplotlib.org/stable/users/index
- pandas: para gestión de bases de datos.
    - https://pandas.pydata.org/docs/user_guide/index.html

Para más información sobre Electrocardiograma:
https://www.my-ekg.com/index.html

Puntos a realizar:

1. Crear código en Python que permita abrir los distintos archivos .csv y .txt para obtener la información de estudios de ECG. Este código debe poder importarse desde un archivo .py que lo ejecute.
2. Crear código para almacenar una Clase Estudio que reúna un único ECG con su detalle. Este código debe poder importarse desde un archivo .py que lo ejecute.
    a. Debe permitir calcular la frecuencia cardíaca en un intervalo de pulsos/datos.
        i. Si no se especifica el intervalo, debe calcular la frecuencia cardíaca promedio.
    b. Debe detectar los latidos (pulsos), de 2 maneras:
        i. Basado en la información del archivo txt.
        ii. Basado en el procesamiento de la señal (búsqueda de máximo relativo). Este apartado lo pueden realizar en la totalidad de la secuencia o para un segmento definido por usuario.
        iii. Basado en alguna librería de Python preexistente.
            1. Grupo 1: neurokit2
                a. https://neuropsychology.github.io/NeuroKit/
            2. Grupo 2: heartpy
                a. https://python-heart-rate-analysis-toolkit.readthedocs.io/en/latest/
            3. Grupo 3: wfdb
                a. https://wfdb.readthedocs.io/en/stable/
            4. Grupo 4: biospy
                a. https://biosppy.readthedocs.io/en/stable/
            5. Grupo 5: py-ecg-detectors (Hamilton)
                a. https://github.com/berndporr/py-ecg-detectors
            6. Grupo 6: py-ecg-detectors (Christov)
                a. https://github.com/berndporr/py-ecg-detectors
            7. Grupo 7: py-ecg-detectors (Engelse and Zeelenberg)
                a. https://github.com/berndporr/py-ecg-detectors
            8. Grupo 8: sleepecg
                a. https://sleepecg.readthedocs.io/en/stable/
            9. Grupo 9: mne
                a. https://mne.tools/1.8/generated/mne.preprocessing.find_ecg_events.html
            10. Grupo 10: python heart rate analysis toolkit
                a. https://python-heart-rate-analysis-toolkit.readthedocs.io/en/latest/heartpy.peakdetection.html

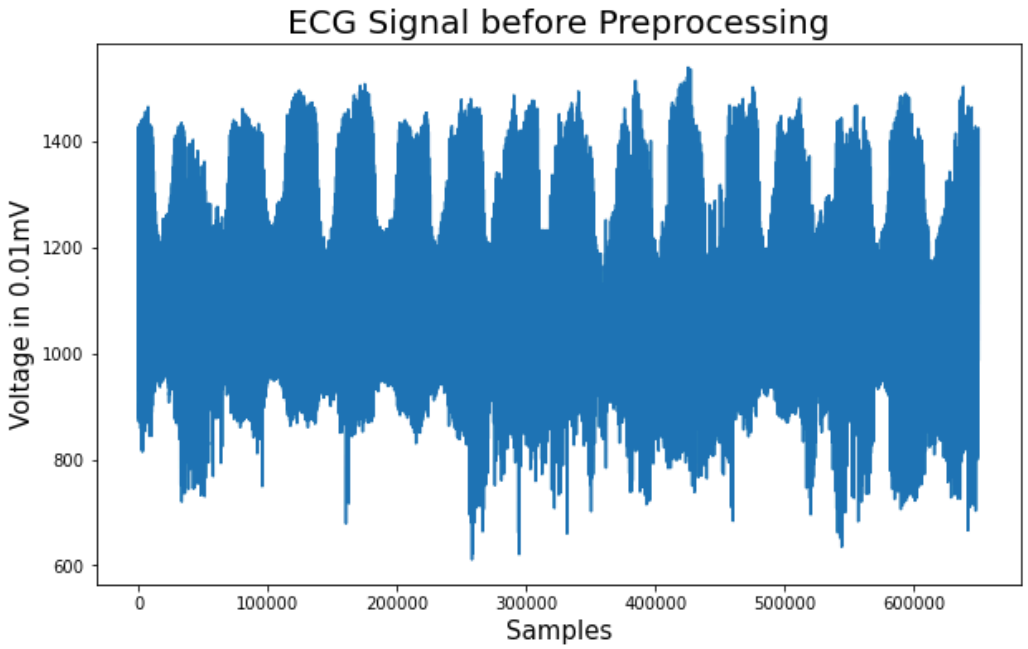
iv. Finalmente, y solo a modo de análisis de su código, comparar cuántos pulsos detecta su código versus lo informado por el archivo de anotaciones.

c. Empleando la librería indicada en el punto b iii, para su grupo, emplear al menos otros 3 métodos en las señales provistas.

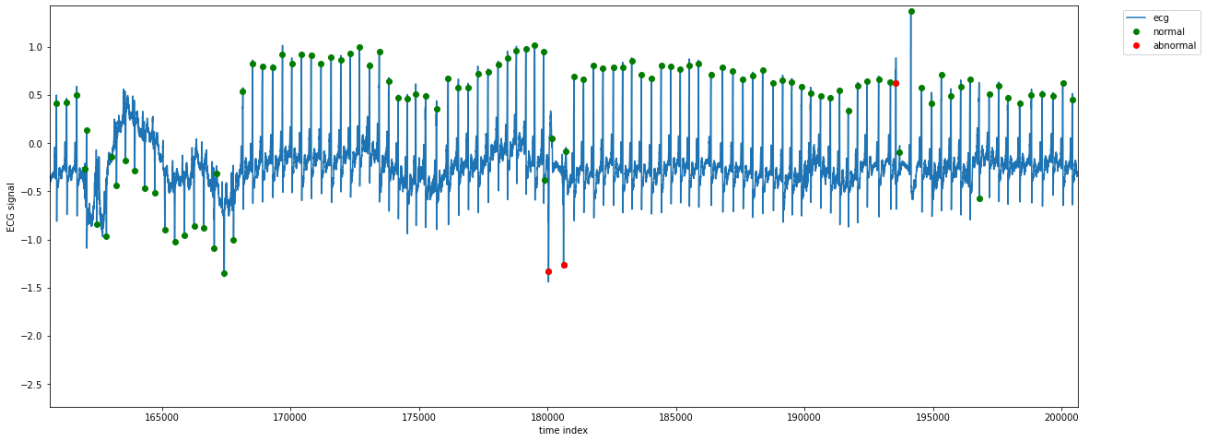
d. Buscar pulsos anormales mediante la información de anotaciones. Esto permitirá, por ejemplo, mostrar en pantalla cuántos pulsos anormales posee la secuencia y en qué momento de la señal se produjeron (ya sea por tiempo o número de muestra).

e. Extraer segmentos del ECG para mostrar por pantalla.

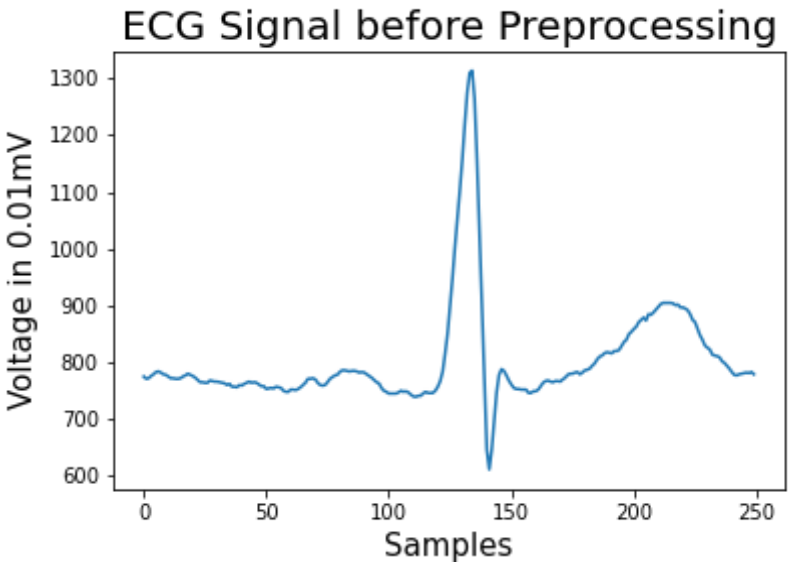
Ejemplo de visualización de señal completa:

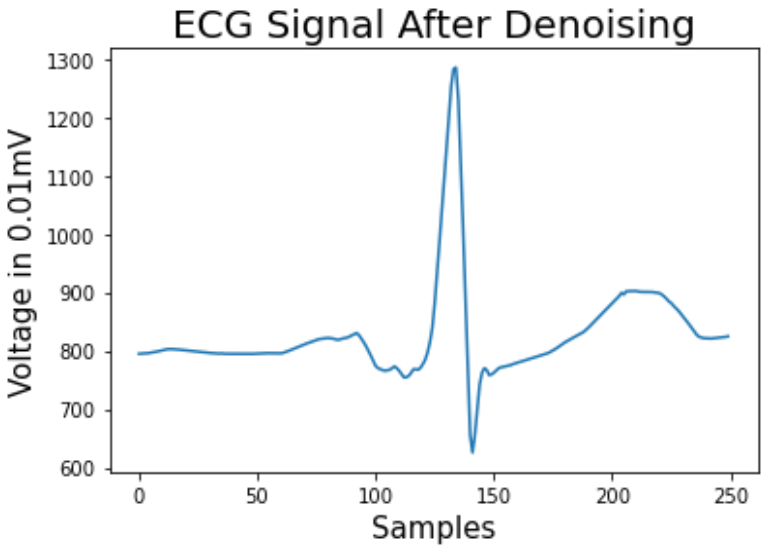


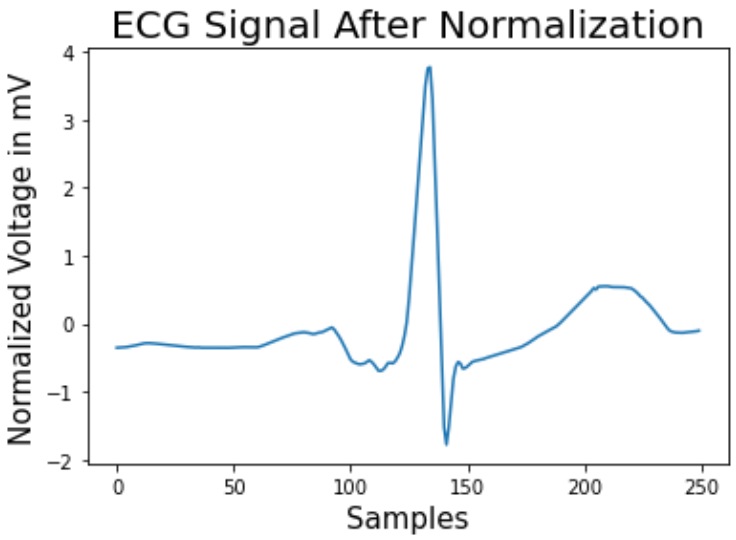
Ejemplo de visualización de segmento (con el agregado de detección de pulsos normales y anormales):



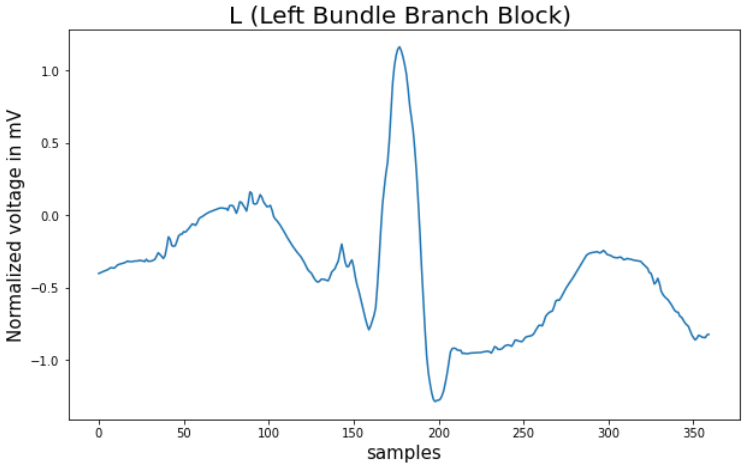
3. Desarrollar un código que permita, a partir del estudio cargado anteriormente en la Clase Estudio, cargar un pulso (latido) en una instancia de Clase Pulso, con las siguientes características:
    a. Atributo que defina el tipo de pulso (normal o anormal, incluye distintas clases de pulso anormal).
    b. Mostrar el pulso en un gráfico.
    c. Normalizar la señal (es decir, la línea de base debe rondar el valor cero).
    d. Suavizar el pulso mediante una media móvil, con la posibilidad de cambiar el ancho de la ventana. Para esto podrá utilizar bibliotecas como “pandas”, combinando métodos como “rolling” y “mean”; u otras herramientas que consideres apropiadas.

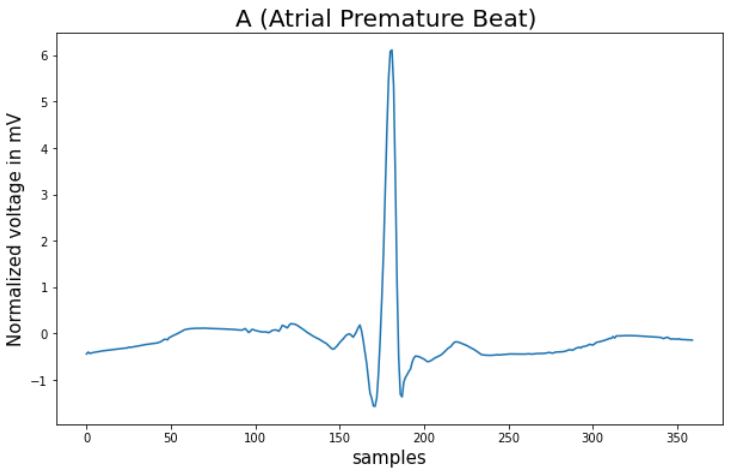






Ejemplos para pulsos anormales:





4. Para completar la introducción a los algoritmos y métodos de inteligencia artificial, cada grupo deberá tratar de ejecutar un Jupyter Notebook (pueden correrlo en Colab o en el IDE que prefieran), con las correcciones que sean necesarias (dado que algunas celdas iniciales de carga de datos pueden no ser necesarias). Cada paso del código deberá ser comprendido y, de ser necesario comentado, para luego explicarlo y defenderlo durante la presentación oral. En caso de que el código no pueda ser ejecutado, lo **fundamental** es que comprendan que acción se realiza en cada momento, su objetivo y su conexión con el resto del procesamiento; dado que se evaluará más la comprensión del código y los pasos, que el hecho de poder ejecutarlo.
    a. Grupo 1 (CNN): https://www.kaggle.com/code/saijithesh/ecg-cnn
    b. Grupo 2 (CNN): https://www.kaggle.com/code/aadesh321kumar/arrythmia-classification#Model-structure
    c. Grupo 3 (SVM): https://www.kaggle.com/code/samaherajili/svm-train-test
    d. Grupo 4 (regresión / ensamble): https://www.kaggle.com/code/unnoun/mit-bih-lr-adaboost-v2
    e. Grupo 5 (SVM): https://www.kaggle.com/code/calulamabel/ecg-classification-normal-v-lbbb
    f. Grupo 6 (CNN): https://www.kaggle.com/code/nasifahmed99/feature-extraction-from-ecg-signal
    g. Grupo 7 (CNN): https://www.kaggle.com/code/khatri007/mitbh-cnn
    h. Grupo 8 (CNN): https://www.kaggle.com/code/behl1anmol/mit-bih-arrhythmia-classification
    i. Grupo 9 (LSTM, KNN, SVM, Tree, Random Forest): https://www.kaggle.com/code/atshayasankaran/classification-of-arrhythmia-lstm-pca-kfold
    j. Grupo 10 (ANN vs DNN): https://www.kaggle.com/code/arkadipmaitra/ann-and-dnn-ecg#ANN