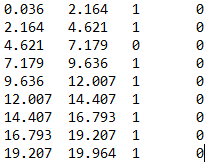
**Trabajo Final Señales**

1. Se descargan los datos ubicados en el siguiente enlace (<https://www.kaggle.com/vbookshelf/respiratory-sound-database>), cada archivo de audio de auscultación (.wav) en su nombre tiene especificado: número del paciente, número de la grabación, localización del pecho donde se realiza la adquisición, modo de adquisición: canal simple (sc) o multicanal (mc) y equipamiento de medición. Cada dato se encuentra separado por guión. Cada uno de estos archivos tiene asociado un archivo de texto con el mismo nombre que posee 4 columnas. Donde la primera corresponde al tiempo de inicio del ciclo respiratorio, la segunda al tiempo de finalización del ciclo, la tercera consta de un 0 o 1 dependiendo si durante el ciclo se presentaron crepitaciones y la última columna también consta de un 0 o 1 para indicar la presencia o ausencia de sibilancia durante el ciclo.



***Figura 1: Archivo de audio y de texto***



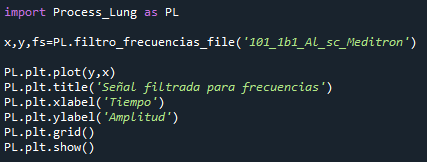
***Figura 2: Columnas del archivo de texto***

1. Dentro de las señales de auscultación, se encuentran embebidas varias señales bilógicas, entre ellas las principales son: las cardiacas y las respiratorias, se sabe que las señales cardiacas se encuentran en un rango de frecuencias de 5Hz a 600Hz, las señales respiratorias están en frecuencias desde los 100Hz hasta 2000Hz, por lo tanto se debe aplicar un pasa bajas de 2000Hz a la señal original y después un pasa altas de 100Hz, para dejar las señales dentro del rango de interés.

Para lograr esto se crea una librería llamada Process\_Lung, dentro de la cual se importan las librerías de soundfile, scipy, pywt y linearFIR. Se crean dos funciones que permiten el filtrado de la señal con los dos filtros mencionados anteriormente. La primera función (filtro\_frecuencias\_file), recibe como parámetro el nombre del archivo .wav entre comillas dobles o simples, después mediante el comando soundfile.read, se extrae del archivo el vector con las muestras de la amplitud y el tiempo de muestreo de la señal, posteriormente se aplica el diseño de filtros que se encuentran dentro del archivo linearFIR, donde se configura un pasa bajas con una frecuencia de corte de 2000Hz y un pasa altas con una frecuencia de 100Hz los cuales se aplican respectivamente al vector de la señal extraída. Con la frecuencia de muestreo y la longitud del vector de muestras se halla el tiempo de la señal y finalmente la función retorna el vector con la señal filtrada, el vector con el tiempo de la señal y la frecuencia de muestreo de la señal original.

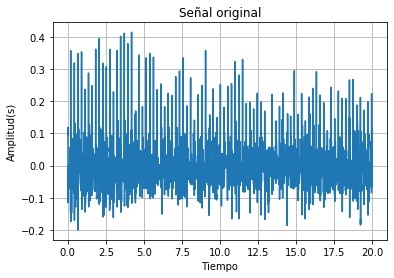
La segunda función (filtro\_frecuencias) recibe como parámetros el vector de la señal original y la frecuencia de muestreo de la señal, posteriormente se aplican las mismas rutinas de la función anterior y retorna las mismas variables.

Se importa la librería Process\_Lung dentro de un nuevo script y posteriormente se aplican los filtros al archivo ‘101\_1b1\_Al\_sc\_Meditron.wav’

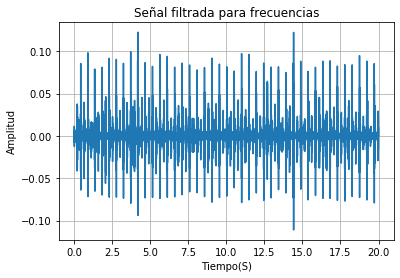


***Figura 3: script para filtrado de la señal de audio***

Algo importante a destacar es que desde el método Process\_Lung, se puede acceder a los métodos de matplotlib.pyplot ya que esta librería se encuentra importada en el script original de Process\_Lung, por lo tanto se puede usar para graficar las señales anteponiendo el método PL. Lo mismo ocurre con las librerías de numpy y pandas.

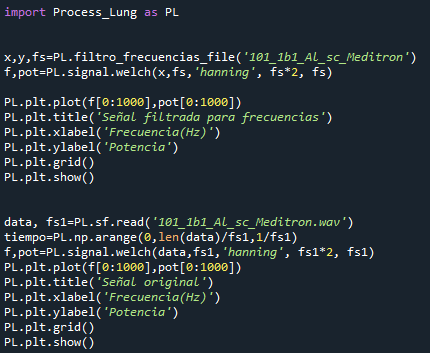


***Figura 4: Señal de audio original***

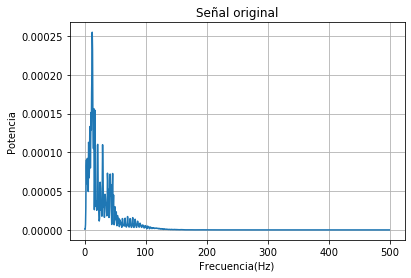
******

***Figura 5: Señal de audio filtrada por frecuencias***

También se pueden extraer los periodogramas de ambas señales usando la librería scipy, para determinar las frecuencias de mayor relevancia para cada señal



***Figura 6: Script para graficar el periodograma de las señales***



***Figura 7: Periodograma-señal original***

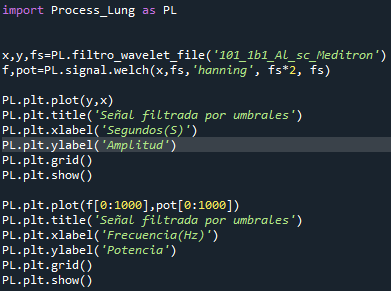


***Figura 7: Periodograma-señal filtrada***

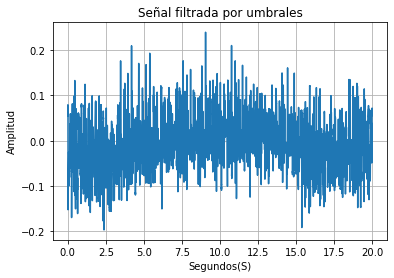
1. Dentro de la librería de “Process\_Lung”, se crean dos funciones que permiten realizar la extracción Wavelet mediante umbrales:

La primera función “filtro\_wavelet\_file” recibe como parámetro el nombre del archivo puesto entre comillas dobles o simples, esta función realiza el filtrado wavelet y posteriormente resta la señal obtenida a la señal original para obtener la señal original umbralizada, esta función retorna esta última señal, el tiempo de la señal y la frecuencia de muestreo.

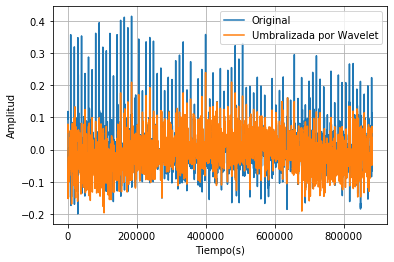
La segunda función “filtro\_wavelet” recibe como parámetros un vector con las amplitudes de la señal y el tiempo de muestreo, esta función también retorna la señal original filtrada mediante umbrales y el vector de tiempo correspondiente de la señal.



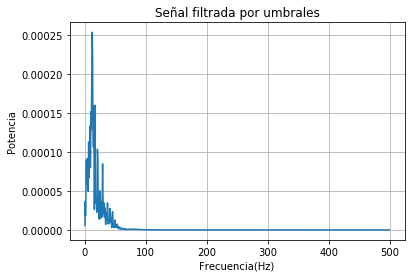
***Figura 8: Script para graficar la señal filtrada por umbrales y su periodograma***

******

***Figura 9: Señal original umbralizada***

******

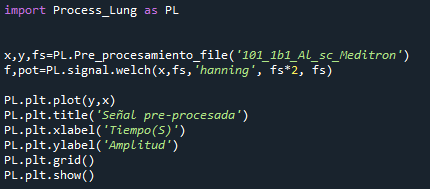
***Figura 10: Señal comparación entre señal original y original umbralizada***

******

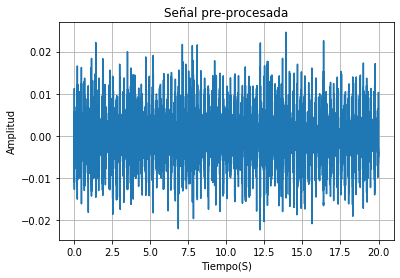
***Figura 10: Periodograma señal filtrada por umbrales***

Al realizar el periodograma para la señal filtrada mediante umbrales usando el filtrado wavelet, se observa que la distribución de las potencias para las frecuencias es idéntica a la que se observa en el periodograma de la señal original (*Figura 7*), por lo tanto, con esta técnica de filtrado por umbrales no se presenta pérdida de información relevante en el dominio de las frecuencias.

1. Para este caso se crea una función dentro de la librería “Process\_Lung”, llamada “Pre\_procesamiento\_file”; esta función recibe como parámetro el nombre del archivo .wav puesto entre comillas dobles o simples y aplica primero la función “filtro\_frecuencias”, explicada en el punto 2 para aplicar un filtro pasa bajas de 2000Hz a la señal y un filtro pasa altas de 100Hz. Posteriormente se usa la función “filtro\_wavelet” explicada en el punto 3, para obtener la función filtrada mediante umbrales y eliminar el ritmo cardiaco, esta función finalmente retorna, un vector con la señal filtrada, el tiempo de la señal y la frecuencia de muestreo.

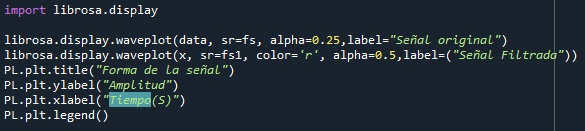


***Figura 11: Script para obtener la señal pre-procesdada***

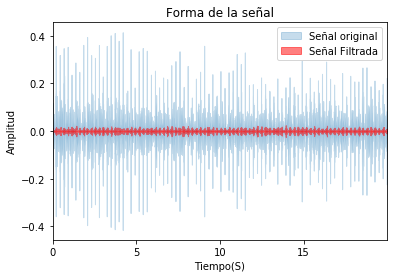


***Figura 12: Señal pre-procesada***

Para comprobar que no hay desfases entre la señal filtrada y la señal original, se importa el módulo display de la librería “librosa”, posteriormente se usa el comando “waveform” para obtener la forma de onda de ambas señales y comprobar que los tiempos de ambas señales coinciden.

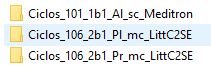


***Figura 13: Script para obtener la forma de onda de las señales pre-procesada y original***



***Figura 14: Forma de onda- Señales coinciden en tiempos***

1. Para la separación de ciclos de la señal de auscultación, se define una función llamada “Ciclos\_respiratorios”; esta función recibe como parámetro el nombre del archivo de texto o de audio entre comillas simples o dobles. Dentro de la función, se ejecutan dos rutinas, la primera que pone al nombre la extensión del archivo .wav para poder extraer el vector de muestras junto con la frecuencia de muestreo usando la librería “soundfile”, también se crea una variable que usa el comando open y que pone la extensión .txt al nombre para así leer el archivo de texto por filas. Posteriormente se crea una carpeta con el mismo nombre del archivo ingresado como parámetro anteponiendo la palabra “Ciclos”(*Figura 15*), dentro de esta carpeta se guardarán las gráficas correspondientes a cada ciclo filtrado.



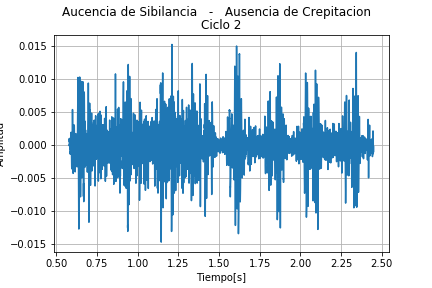
***Figura 15: Carpeta creada mediante función***

La segunda rutina identifica los datos del archivo de texto por columnas y guarda cada columna como vector mediante un ciclo “for”. Luego mediante otro ciclo “for” se busca en el vector de tiempo de la señal los valores que coincidan para los tiempos de inicio y fin del ciclo respiratorio reportado en el archivo de texto, después se construye el vector de ciclo con estos tiempos y lo mismo se hace con el vector de tiempo. Posteriormente se filtra la señal mediante la función “Pre\_procesamiento” explicada en el punto 4, se grafica el ciclo y mediante condicionales se colocan súper títulos que indican si el ciclo posee ausencia o no de crepitaciones o sibilancias y se guarda en la carpeta creada inicialmente.



***Figura 16: Graficas de ciclos guardados mediante ciclo for***

Finalmente la función retorna 4 vectores, el primero contiene los ciclos de la señal, donde en cada posición se guardan todos los datos del ciclo, el segundo contiene los vectores de los tiempos correspondientes a los ciclos, el tercer vector posee un arreglo donde en cada posición se guarda un “1” o “0” dependiendo de si el ciclo correspondiente posee crepitancia o no, el último vector también guarda en sus posiciones un “1” o “0” dependiendo de si se presentan o no sibilancias y finalmente se retorna un escalar que corresponde a la frecuencia de muestreo de la señal.



***Figura 17: Ejemplo de ciclo obtenido mediante la función descrita.***

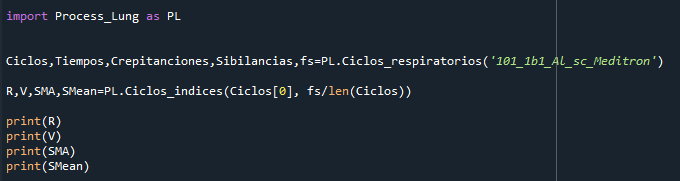
1. Para extraer los índices requeridos, se definen 4 funciones dentro de la librería “Process\_Lung”. La primera función “Rango”, recibe como parámetro el vector del ciclo que se quiere analizar, posteriormente halla el máximo y el mínimo del vector mediante la librería “numpy” y extrae el valor absoluto de la resta para obtener el rango estadístico de la señal.

La segunda función “SMA”, también recibe como parámetro el vector del ciclo de interés y mediante un ciclo “for” crea un arreglo que contiene la resta de la posición Ciclo(n+1)-Ciclo(n), posteriormente mediante np.sum() se realiza la sumatoria de todos los datos ingresados y este valor se retorna como el promedio de la media móvil simple.

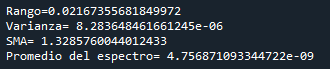
La tercera función “Spect\_mean” recibe como parámetros el vector con el ciclo de interés y la frecuencia de muestreo, posteriormente se realiza el periodograma usando el comando “welch” de la librería scipy y finalmente retorna el promedio de las potencias calculadas.

Para hallar la varianza simplemente se usa el comando np.var() y se ingresa como parámetro el vector que contiene el ciclo de interés.

Ahora, también se crea una función llamada “Ciclos\_indices” que recibe como parámetros el ciclo de interés y la frecuencia de muestreo de la señal, en su interior se llaman las funciones antes explicadas y retorna los cuatro índices estadísticos.



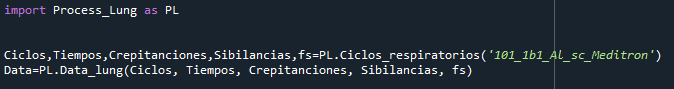
***Figura 18: Script para extraer los ciclos con sus respectivos índices estadísticos.***



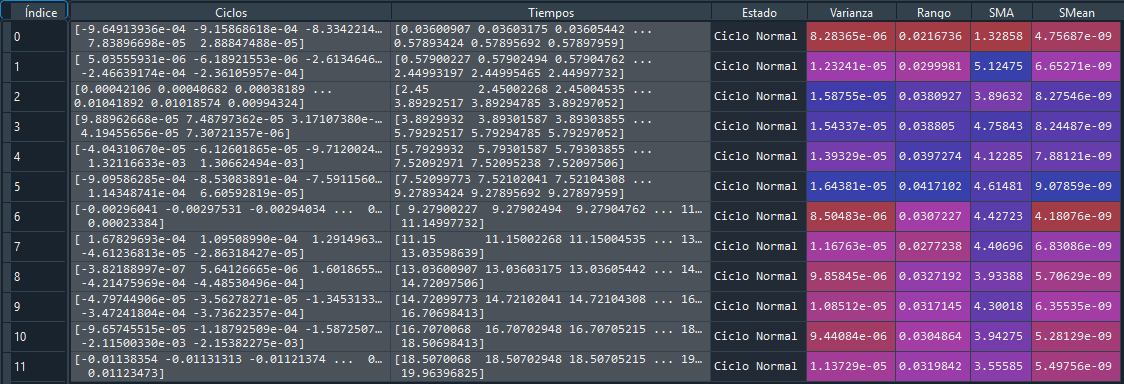
***Figura 19: Resultado para los índices calculados***

1. Se crea una función “Data\_Lung” que recibe como parámetros los valores retornados por la función “Ciclos\_respiratorios” es decir: el vector con los ciclos, el vector con los tiempos de los ciclos, el vector con la presencia o no de crepitaciones, el vector con la presencia o no de sibilancias y la frecuencia de muestreo.

La función crea un dataframe con los siguientes campos: “Ciclos”, “Tiempos”, “Estado”, “Varianza”, “Rango”, “SMA” y “SMean”. Inicialmente las únicas columnas completas son las correspondientes a la de “Ciclos” y “Tiempos”, posteriormente mediante un ciclo “for” para cada ciclo de la señal calcula mediante condicionales si hay presencia o ausencia de crepitaciones o sibilancias y según el resultado llena la columna de “Estado” con cadenas que pueden ser: “Ciclo normal”, “Ciclo con Crepitación” y “Ciclo con Sibilancia”, también se completan las demás columnas para los índices usando las función “Ciclos\_indices”.

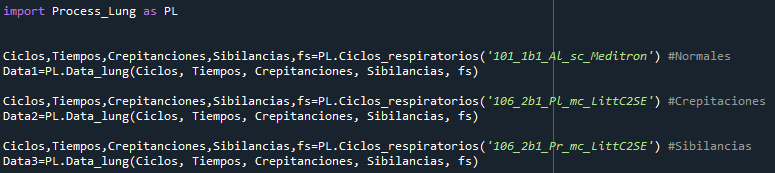


***Figura 20: Script para obtener el dataframe de los ciclos de una señal***

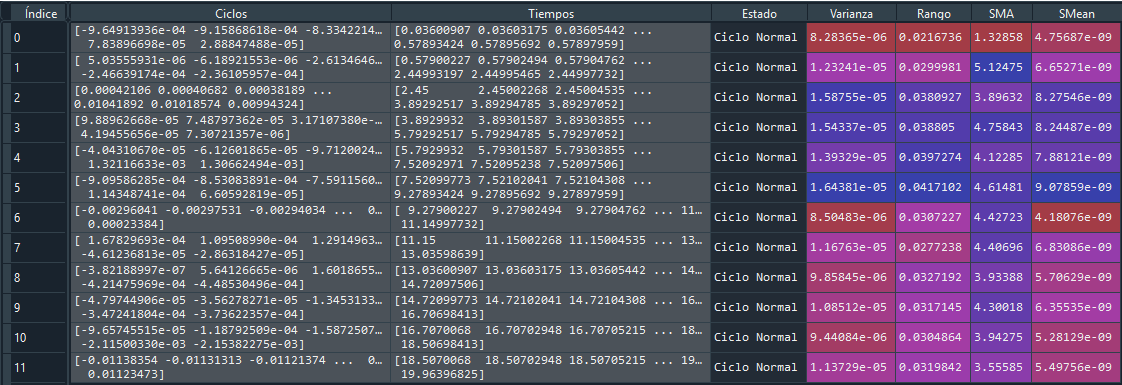


***Figura 21: DatasFrame obtenido***

1. Para realizar el análisis estadístico se usaron tres archivos de la base de datos con sus respectivos archivos de texto, el primero archivo (‘101\_1b1\_Al\_sc\_Meditron’) posee principalmente ciclos normales con ausencia de sibilancias y crepitaciones, el segundo (106\_2b1\_Pl\_mc\_LittC2SE), posee ciclos principalmente con crepitaciones y el tercero (106\_2b1\_Pr\_mc\_LittC2SE), posee ciclos principalmente con sibilancias.



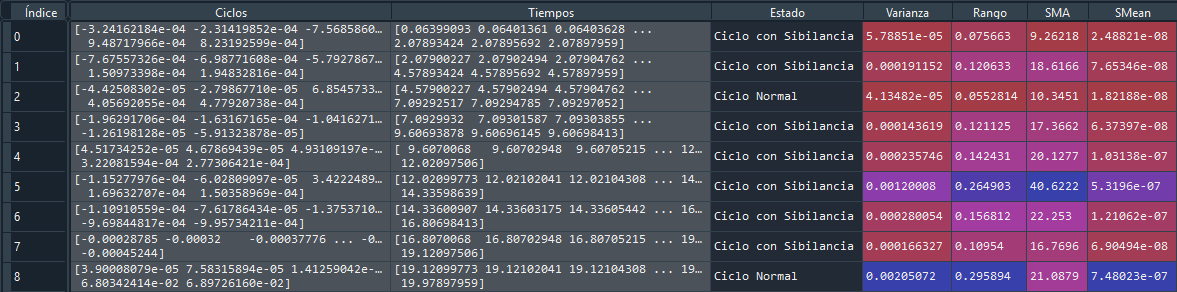
***Figura 22: Adquisición de los DataFrame***



***Figura 23: DataFrame de ciclos normales***



***Figuras 24: DataFrame de ciclos con Crepitaciones***



***Figura 25: DataFrame de ciclos con Sibilancias***

**Análisis:**

Al analizar detenidamente los DataFrames, se observa que para los ciclos normales el SMA no supera el valor de 10, sin embargo, al observar los ciclos con sibilancias o crepitaciones, este valor varía desde los 11 hasta los 35 o 40, y los demás índices estadísticos como la varianza y el promedio del espectro siguen siendo similares. Ahora, los ciclos con sibilancias presentan un valor mucho mayor para los rangos en cada uno de sus ciclos siendo superiores a los 0.1 con respecto a lo observado en los ciclos normales y con crepitaciones donde los valores no superan los 0.04 en promedio.