Name: **Date:** **Class:**

Downsampling Code Explanation – For the Teacher

Downsampling is a technique used to reduce the number of data points in a dataset. This is particularly useful when dealing with large datasets, such as EMG signals, to make them more manageable and to speed up the analysis. Downsampling reduces the **sampling frequency** by selecting fewer data points while maintaining the integrity of the overall pattern or trend. This is often done by skipping over certain samples in the original dataset.

Why is Downsampling Important?

- **Data Reduction**: It reduces the data size, making it easier to process and analyze.
- **Noise Reduction**: Downsampling can help reduce noise by smoothing out small variations that may not be relevant for the analysis.
- **Faster Processing**: Smaller datasets require less computational power, which is beneficial when performing analyses such as machine learning or statistical testing.

How Downsampling Works

When we downsample, we essentially keep data points from the original signal at a reduced frequency. For example, if we start with a signal sampled at 44,100 Hz (samples per second), and we downsample it to 8,000 Hz, we keep every fifth data point to match the new frequency.

Downsampling Script Explanation

The following Python script demonstrates how to downsample data using **SciPy** and **NumPy**. The students will be working with this script to apply the downsampling technique to their EMG data.

Python Script for Downsampling (In the Jupyter Notebook)

```
import numpy as np
from scipy.io import wavfile
import matplotlib.pyplot as plt
# Downsample to 8000Hz
downsampled_data = resample(data, int(len(data) * (8000 /
sample_rate)))  # Resample the data to 8000 Hz
duration_downsampled = len(downsampled_data) / 8000  # Calculate the
duration of the downsampled audio
time_axis_downsampled = np.linspace(0, duration_downsampled,
len(downsampled_data))  # Create a time axis for the downsampled data
df_downsampled = pd.DataFrame({'Time (seconds)': time_axis_downsampled,
'Amplitude': downsampled_data})  # Create a DataFrame with downsampled
time and amplitude
df_downsampled.to_csv('/content/drive/MyDrive/Colab
Notebooks/outputFile_downsampled.csv', index=False)  # Save the
downsampled DataFrame as a CSV file
```

Explanation of the Script:

```
downsampled_data = resample(data, int(len(data) * (8000 / sample_rate)))
# Resample the data to 8000 Hz
```

- **Explanation**: This line uses the *resample* function (likely from scipy.signal or another library) to downsample the data.
    - **data**: This is the original signal that was loaded (e.g., a muscle signal from the .wav file).
    - **sample_rate**: The original sampling rate of the data (for example, 44,100 Hz).
    - **len(data)**: This returns the number of data points (samples) in the original signal.
    - **(8000 / sample_rate)**: This is the ratio of the target sampling rate (8000 Hz) to the original sampling rate. This ratio determines how many samples should be kept in the downsampled signal.
    - **int(len(data) * (8000 / sample_rate))**: The total number of samples to keep after downsampling. This calculates the new length of the signal based on the target rate of 8000 Hz.
    - **resample()**: This function resamples the original data to the target number of samples, effectively reducing the frequency.
    - **Result**: The downsampled_data now holds the resampled signal at 8000 Hz.

```
duration_downsampled = len(downsampled_data) / 8000  # Calculate the
duration of the downsampled audio
```

- **Explanation**: This line calculates the **duration** (in seconds) of the downsampled audio signal.
    - **len(downsampled_data)**: This returns the number of samples in the downsampled signal.
    - **8000**: This is the new sampling rate (8000 samples per second).
    - **len(downsampled_data) / 8000**: The duration of the downsampled audio is calculated by dividing the number of samples in the downsampled signal by the sampling rate (8000 samples per second).
    - **Result**: The *duration_downsampled* variable now contains the total time (in seconds) of the downsampled audio.

```
time_axis_downsampled = np.linspace(0, duration_downsampled,
len(downsampled_data))  # Create a time axis for the downsampled data
```

- **Explanation**: This line generates a **time axis** for the downsampled data to plot the signal or to represent the time associated with each data point.
    - **np.linspace(0, duration_downsampled, len(downsampled_data))**: This generates an array of evenly spaced time values from 0 to duration_downsampled, with the same number of points as there are in the downsampled data.
        - **0**: The start time of the signal (0 seconds).
        - **duration_downsampled**: The end time of the signal, calculated in the previous line.

        - **len(downsampled_data)**: This ensures the time array has the same number of elements as the downsampled data, so each data point is associated with a corresponding time value.
    - **Result**: *time_axis_downsampled* is an array that represents the time in seconds for each sample in the downsampled data.

```
df_downsampled = pd.DataFrame({'Time (seconds)': time_axis_downsampled,
'Amplitude': downsampled_data})  # Create a DataFrame with downsampled
time and amplitude
```

- **Explanation**: This line creates a **Pandas DataFrame** that combines the time and amplitude of the downsampled signal into a tabular format.
    - **pd.DataFrame()**: This function creates a DataFrame (a table-like structure in Python) from the provided dictionary.
        - **'Time (seconds)'**: This is the name of the column that will store the time values.
        - **time_axis_downsampled**: The time values (generated in the previous line).
        - **'Amplitude'**: This is the name of the column that will store the amplitude (signal strength) values.
        - **downsampled_data**: The amplitude values of the downsampled signal.
    - **Result**: *df_downsampled* is a DataFrame where the first column contains the time values, and the second column contains the corresponding amplitude values of the downsampled signal.

```
df_downsampled.to_csv('/content/drive/MyDrive/Colab
Notebooks/outputFile_downsampled.csv', index=False)  # Save the
downsampled DataFrame as a CSV file
```

- **Explanation**: This line saves the **downsampled data** in .csv format to Google Drive (or another location).
    - **df_downsampled.to_csv()**: This function saves the DataFrame (df_downsampled) as a .csv file.
        - **'/content/drive/MyDrive/Colab Notebooks/outputFile_downsampled.csv'**: This is the file path where the .csv file will be saved. In this case, it is saved to Google Drive within a folder named Colab Notebooks. Students can modify the path to save the file to a different location if desired.
        - **index=False**: This prevents Pandas from saving the index (row numbers) as a separate column in the .csv file.
    - **Result**: The downsampled data is saved as a .csv file named *outputFile_downsampled.csv* that students can open in a program such as Excel or import into other analysis tools.

Name: Date: Class:

Summary of what happens in the script:

- Downsampling: The original signal is downsampled from its original rate to 8000 Hz.
- Time Axis Creation: A time axis is generated to correspond to the downsampled data.
- Data Storage: A Pandas DataFrame is created, combining the time and amplitude of the downsampled signal.
- Saving the Data: The DataFrame is saved as a .csv file, making it accessible for further analysis or visualization.

Patterns in Downsampled Data

- **What Students Will See:**
    - **Simplified Signal**: The downsampled data will retain the key features of the signal but with fewer data points.
    - **Major Trends**: Students can identify broad patterns of activity without the distraction of very fine-grained detail.

- **Relation to Muscle Activity and Movement**:
    - Downsampling highlights overall trends in muscle activity, making it easier to compare across datasets or detect long-term behaviors (e.g., fatigue over time).
    - It may slightly obscure finer details, but the focus remains on significant movements.