I started by looking at research papers, with the main paper that I drew inspiration from being,

**V. Zakeri and A. J. Hodgson, "Automatic Identification of Hard and Soft Bone Tissues by Analyzing Drilling Sounds," in IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 27, no. 2, pp. 404-414, Feb. 2019, doi:10.1109/TASLP.2018.2880336.**

This paper provided a guide of what preprocessing needed to be used for acoustic signals, what features could be drawn from these signals, and different methods to classify these signals.

**Key Topics:**

- **Preprocessing and Windowing:**

- Sampling Frequency

- Windows should be small to provide feedback in close to real time

- Filters

- **Feature Extraction**

- Short-time Fourier transform coefficients (STFTCs)

- Mel-frequency cepstrum coefficients (MFCCs)

- Wavelet packet transform coefficients (WPTCs)

- Can also look into using audio features

- **Identification methods**

- Logistic regression (LR)

- Support vector machine (SVM) (RadialBasisFunction (RBF) Kernel)

- Random forest (RF)

- Hidden Markov model (HMM)

- The strongest identification methods were SVM-RBF and RF; and the best features were WPTC and STFT (Gonna start by investigating these)

I then researched basic sound signal processing to determine similarity between two signals:

- cosine similarity - calculates the angle between two features → have to find vectors that can be compared

- correlation coefficients

- pearson correlation- measures the linear relationship between two continuous variables

- euclidean distance - distance between corresponding points in a time signal

- Cross correlation- shifting signals and multiplying them together to see when they match up

- Dynamic time warping - compares sequences of data with different lengths. Useful when comparing signals that may vary in speed or timing.

I then moved on to testing these similarity calculations with sound signals I produced by tapping on wood and tapping on metal. These results are displayed on my research poster, which is attached in the github.

The most promising similarity comparison was dynamic time warping when applied to STFTs.

Code Breakdowns:

MFCC-similarity.py - provides similarity scores for the MFCCs of different signals.

Wavelet-level-plot.py - provides wavelet decomposition of a sound signal along with visualizations.

centroid.py - provides similarity scores for the spectral centroid of a sound signal.

Discrete\_ft.py - provides the discrete fourier transform of a sound signal.

display-wav.py - displays the waveform plot of a sound signal.

**dtw-promising.py** - provides the DTW distance of the STFTs of two sound signals.

dumbcrosssorr.py - provides the cross correlation of two sound signals.

filtered-display.py - provides a visualization of a sound signal before and after passing it through a high pass filter.

filtered-low-display.py - provides a visualization of a sound signal before and after passing it through a low pass filter.

filtered-dtw.py - provides the DTW distance of the STFTs of two sound signals after they have been passed through a high pass filter.

stft-all-comparisons.py - provides similarity scores based on Mean Squared Error, Pearson Correlation Coefficient, and Euclidean distance.

stft-display-materials.py - provides STFT spectrograms

stft-display-p2.py - provides STFT spectrograms

wpt-all-comparisons.py - provides similarity scores based on wavelet packet transform coefficients