Features.csv

Essentially, the way they created the ~500 columns of the features comes from running every function on the music analysis package “libROSA” on the given track, then computing the mean, standard deviation, skew, kurtosis, median, min, and max of that output. Therefore, each output from libROSA gives us at least 7 columns, and more on the functions that already output multiple columns. These functions are as such:

chroma\_stft:

Output: 12 columns, one for each note (there are 12 notes in an octave)

End result: 12 x 7 = 84 columns

Starts out analysis with a fourier transform. Essentially finds what notes are playing at any given time. They use frame lengths of 2048 time units and skips of 512 time units (notice this means each frame overlaps significantly with its neighbors).

Intuitive understanding: the mean of the “G” column would be “How often/with what intensity is the note G being played in this song, in any octave (all functions use a range of 7 octaves, which should be enough for any average piece of music)?”

Usefulness to us: Basically none. How often a note is played says nothing about genre

chroma\_cqt:

Essentially chroma\_stft, but with a “Constant Q Transform” instead of a fourier transform.

chroma\_cens:

Same as above, but normalized.

tonnetz:

Output: 6 columns, Fifth x-axis, Fifth y-axis, Minor x-axis, Minor y-axis, Major x-axis, Major y-axis

End Result: 6 x 7 = 42 columns

Takes as input one of the “chroma” functions’ outputs and creates kind of a relationship between the tones. I think the way this works is if a major chord is played, that goes into “Major y-axis” and if a major arpeggio is played, that goes into “Major x-axis”? Their documentation on this isn’t super clear, but I think it works this way for Major, Minor, and Dominant chords.

Intuitive understanding: the mean of the “Minor y-axis” column would be “how much/with what intensity are block minor chords played in the song?”

Usefulness to us: Reasonably important. Different genres will use different chords.

mfcc:

Output: 20 columns, one for each “coefficient”

End result: 20 x 7 = 140 columns!

Their code first creates a mel-scale from the frequencies (it’s like a weird log scale that comes from a survey of human beings and is supposed to represent how well humans hear the differences in pitches?) and then it puts it through a ton of weird math that I don’t understand to get some sort of meaningful coefficients.

Intuitive understanding: Here’s some info: <https://en.wikipedia.org/wiki/Mel-frequency_cepstrum> If you manage to understand what’s going on, let me know. Apparently it’s very useful in doing voice-to-text stuff.

Usefulness to us: I don’t know why, but very important, according to the algorithm.

rmse:

Output: 1 column, root mean squared energy

End result: 1 x 7 = 7 columns

Takes in a spectrogram and calculates the root mean squared energy. I think this means it like tells you how loud the music is at different times. Seems pretty intuitive to me, if that’s correct.

Usefulness to us: somewhat, as different genres may have different volume profiles.

zcr:

Output: 1 column, zero-crossing rate

End result: 1 x 7 = 7 columns

I don’t know what it really means to “cross zero” in an audio sample, but I read that this is often used to “classify percussive sounds.” If that’s true, type of percussion is important to the genre of music and we should keep it in mind.

Usefulness to us: algorithm says not incredibly important, but still useful, as expected.

spectral\_centroid:

Output: 1 column, centroid of the spectrogram at each frame.

End result: 1 x 7 = 7 columns

It seems that for each frame of the stft we computed, they take the mean, so you essentially know where the “pitch center” of the song is at any given moment. Uses the same 2048, 512 sampling as before.

Usefulness to us: maybe not very important. Does it matter what pitch the sample is centered around?

spectral\_bandwidth:

Output: 1 column, standard deviation of the spectrogram at each frame.

End result: 1 x 7 = 7 columns

Looks like they take the standard deviation of the stft at every frame, so it’s essentially “how open is the chord?”

Usefulness to us: Quite useful. Certainly some genres use wider pitches than others

spectral\_contrast:

Output: 6 columns, one for each octave

End result: 6 x 7 = 42 columns

From what I can see, first the stft is split into 6 octave-length subspectra, then the difference between the peaks and valleys is analyzed. <http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=C570D8446DB5449A45052136A85349E9?doi=10.1.1.583.7201&rep=rep1&type=pdf>

This paper gives some insight and compares it against MFCC which it says simply “averages each subspectrum” and therefore isn’t as good at classification.

This quote from the paper is especially relevant for the intuition:

“For most music, the strong spectral peaks roughly correspond with harmonic components; while non-harmonic components, or noises, often appear at spectral valleys.”

Essentially, I think it looks at the differences between he harmonic and non-harmonic components of the music in each octave.

Usefulness to us: Top. Outperforms everything in the analysis and makes sense as to why it’s important.

Spectral\_rolloff:

Output: 1 column, rolloff at each frame

End result: 1 x 7 = 7 columns

Seems to be just at what rate the highs and lows of the song get softer or louder. Looks mostly meaningless.

Usefulness to us: low. Analysis agrees that it’s not very useful.