Niche Music Genre Classification

Author: Cael Lohmiller

Abstract:

The GTZAN Dataset is a machine learning dataset for music genre classification. The problem with it is the genres are very broad and I argue that most people who listen to music can identify the genres used in the GTZAN dataset (Blues, Classical, Country, Disco, HipHop, Jazz, Metal, Pop, Reggae, Rock). It is also missing major genres! (R&B for example). Therefore, there is limited utility in having a good model using the GTZAN dataset. What I wanted to do create a dataset that has more niche music genres that are more difficult for a human to identify, thus providing more utility.

Creating a dataset is not always a simple task. I really don't know how the creators of the GTZAN dataset can claim they worked on it for 3 days. 3 DAYS. It took me at least a week to even figure out what genres I was going to try classifying. There are apparently hundreds of niche music genres for every 1 genre the GTZAN dataset used according to rateyourmusic.

I lost sight of my goal in the middle of trying to create the dataset because in my mind, the first step to my project is having the song files. Really, I could have done a lot of the pre-processing work separately from the goal of getting the dataset, ie. I didn't have to get every song file before I could do some audio processing.

I have nearly achieved the goal of creating a new dataset, but that was not the whole reason I started this project. I wanted to do some serious audio processing and machine learning! Having the dataset in my hands but it not being properly processed like the GTZAN dataset so it can be used for machine learning leaves a bitter taste in my mouth.

Introduction:

The problem I had set in my mind from the beginning was making a new dataset that is better than the GTZAN dataset by being more useful to people who love music. By people who love music, I mean those who know they love one genre more than others and go out of their way to find interesting examples of that genre. That is where I personally get the most gratification is finding a weird or unexpected new artist that is so unique because it is not following the general idiosyncrasies of their genre. When you find that niche section of music, you just want more. What better way to find more than classifying what you have found and looking for other artists that have also been classified as that. Imagine hearing a metal artist that you click with completely and not knowing that it's metal music because metal music doesn't exist, and then you pull out your phone and it tells you it's a mix between classical and blues music. That is what makes this project fun and exciting.

The results of my project just come down to me having the dataset that I needed to do the rest of the project. 100 songs for 13 "niche" genres within electronic music. That totals to 1300 songs. I put niche in quotations because the genres I picked are the most popular genres of electronic music, they are just niche in comparison to electronic music as a whole. The song files are mp3 files, and I wanted them to be a different bitrate and length at the time of stopping.

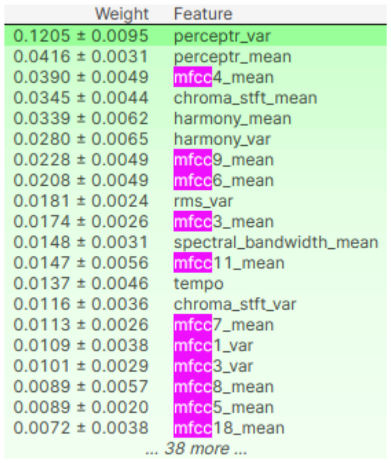
Related Work :

The GTZAN dataset was “collected in 2000-2001 from a variety of sources including personal CDs, radio, microphone recordings, in order to represent a variety of recording conditions”. To that I would say I have no idea how they did this, or how long it took. It couldn’t have been easy. I’m one person and it seems like they did it with 4 people, so that helps explain. Nowadays, I imagine getting recordings from the radio or microphones is unnecessary because we can distort the audio files we have to help generalize the model.

Their approach is very similar to mine, all that differs is what genres we are classifying and therefore what song files are being used. It’s not like I could have just replaced all their song files, used the top Kaggle code and called it a day because they don’t just use the song files for the machine learning, there is lots of post-processing that comes with the dataset itself.

Data:

The data I used to create the dataset was mainly text and html. To work with the dataset, there were post-processing values such as these:



Weight	Feature
0.1205 ± 0.0095	perceptr_var
0.0416 ± 0.0031	perceptr_mean
0.0390 ± 0.0049	mfcc4_mean
0.0345 ± 0.0044	chroma_stft_mean
0.0339 ± 0.0062	harmony_mean
0.0280 ± 0.0065	harmony_var
0.0228 ± 0.0049	mfcc9_mean
0.0208 ± 0.0049	mfcc6_mean
0.0181 ± 0.0024	rms_var
0.0174 ± 0.0026	mfcc3_mean
0.0148 ± 0.0031	spectral_bandwidth_mean
0.0147 ± 0.0056	mfcc11_mean
0.0137 ± 0.0046	tempo
0.0116 ± 0.0036	chroma_stft_var
0.0113 ± 0.0026	mfcc7_mean
0.0109 ± 0.0038	mfcc1_var
0.0101 ± 0.0029	mfcc3_var
0.0089 ± 0.0057	mfcc8_mean
0.0089 ± 0.0020	mfcc5_mean
0.0072 ± 0.0038	mfcc18_mean
… 38 more …	

Mfcc stands for Mel-Frequency Cepstral Coefficients, which help describe the spectral envelope. Harmony refers to harmonics. Percept refers to the perceptrual understanding shock wave that represents the sound rhythm and emotion. Tempo is the beats per minute. Chroma refers to chroma features are a represention the entire spectrum projected onto 12 bins representing the 12 distinct semitones of the musical octave. These are abstract musical characteristics put into numbers that are hard to wrap your head around sometimes. The ear gives us the sense of hearing and allows us to identify genres. Putting that process into mathematical terms is difficult, and the things that make genres very different to us for example Electronic music uses synthesizers and Rock music uses guitars are obviously not as simple to a computer. Maybe if there was a mathematical way to tell what instruments are within each song, that would help the model but I don’t see any of those being used in the top GTZAN solution.

The data used to create these quantities is the pure sound files. The amount of data is very large because there are 100 songs for each genre used and each song has about 22050 KHz with an extremely high bit rate. The reason the bit rate is so high (as far as I can tell) is the GTZAN dataset is using .wav files which are lossless. I wanted to use lossy audio files. The GTZAN dataset decides to half the sample rate which makes everything sound muddy, and then I make the target bitrate of my dataset to be 64 kbps at 48000 KHz and it sound pretty much the same. It is very confusing to be to be honest because the GTZAN dataset has such a large file size per second in comparison to my dataset when they sound the same to my ears. Why doesn't the GTZAN dataset sound better than my dataset when its file size is so much larger? Was this a conscious choice? Why didn't they lower the file size to be as small as possible without compromising sound quality? I'm missing something here.

It makes sense to use low resolution audio, and to keep the audio to 30 second clips. This makes the processing more manageable, and it is debatable how much that compromises the accuracy of the model.

Methods:

To create my dataset, I started by finding the most popular genres within electronic music. To find the most popular genres, I based it off of the number of releases for each of the genres. To get that number for every genre, I had to go to 357 subgenre webpages that exist in electronic music on rateyourmusic because they don't give you the release number for every genre on one webpage. Why did I use rateyourmusic? Because there is no other music rating website that is as popular. It is important for it to be popular, if it wasn't popular then you would have 2 people voting to say an album exists in a specific genre, which lowers the credibility of this entire process (creating the niche genre database).

Once I had the most popular genres, I moved on to getting the chart/leaderboard for each genre. To do that, I decided to sort the leaderboard by most popular all time (for each genre). Once again, it is important to go by popular to make the genre classification of the website itself the most accurate it can possibly be.

Rateyourmusic is mainly for album charts. Song charts are not as popular. It is popular to rate albums and then rate the songs in relation to the album, not rate songs in and of themselves.

I then found the albums for each genre chart by copying the html of each webpage. I scraped through the html to find the album links so that I can automatically move from one album to the next and take the information from each album webpage.

The reason I needed to go to each album webpage is to select the song I want to use for the database. This was done by scraping for the album ratings from each webpage. This process had to be automated because it would take me literally forever to find the 1300 songs from each album page manually by hand. So it was all done with python keyboard automation, for example making python press alt-tab, then press ctrl-a ctrl-c to copy, etc.

Easy peasy? Sort of. I was also scraping to see if I was being blocked from accessing the webpage because I loaded to many webpages so that I could complete the human verification and keep scraping for information. All of this would have been unnecessary if there the api that is planned for this website existed yet. I got my program to also fix its mistakes. Sometimes the program didn't find the album it was looking for and kept going, leaving a missing slot in my text file. I programmed it to research the websites it missed if any.

Then, I created a youtube playlist for every genre once I knew what songs I needed. I made this faster with some more python automation copying and pasting from files into youtube.

I guess the playlists are the best thing to come out of this entire experience :(. Oh well.

https://www.youtube.com/playlist?list=PLFVFsGE1PWSeJAbWma9x036Aln9UFZphA

https://www.youtube.com/playlist?list=PLFVFsGE1PWScJixZ_SpmsJ32chndz0mO0

https://www.youtube.com/playlist?list=PLFVFsGE1PWSeXk_XtQEHsTg2ly72iOJsz

Video:

https://drive.google.com/file/d/1id87h27gThGaqdOmgPyToHv2PXg_oNqX/view?usp=sharing