**What is goal of this project?**

Build a predictive system or a machine learning model to classify the genre of a given song, i.e., if a song with a bunch of different attributes is fed into the machine learning model, the model should classify the category to which the song belongs.

**Steps to build music genre classification model**

Classifying music genres used to be a complicated and time-consuming process, but machine learning has made it possible to do the task in a matter of seconds. So let's take a look at the steps required to create a machine learning model that classifies music.

Diagram

Description automatically generated

**About Dataset**

This dataset was used for the well-known paper in genre classification "Musical genre classification of audio signals" by G. Tzanetakis and P. Cook in IEEE Transactions on Audio and Speech Processing 2002.

There are some practical and conceptual issues with this dataset, described in "The GTZAN dataset: Its contents, its faults, their effects on evaluation, and its future use" by B. Sturm on arXiv 2013.

The lesson is not to banish GTZAN, but to use it with consideration of its contents.

Content The dataset consists of 1000 audio tracks each 30 seconds long. It contains 10 genres, each represented by 100 tracks. The tracks are all 22050 Hz monophonic 16-bit audio files in .au format.

**extract\_features():**

The function uses the librosa library to load the audio file and extract three types of features from it:

* loads an audio file located at file\_path using the librosa.load function. The function returns two values: the audio signal, represented as a one-dimensional NumPy array, and the sampling rate of the audio file, which is **the number of samples per second used to represent the signal**. These values are then assigned to the variables signal and sr, respectively.
* **Mel Frequency Cepstral Coefficients (MFCCs):** These are commonly used features for audio processing and are obtained by computing the Discrete Cosine Transform (DCT) of a log-magnitude mel-scale spectrogram of the audio signal. In this code, we compute the MFCCs using the librosa.feature.mfcc function.
* **Spectral Centroid**: This is a measure of the "center of mass" of the frequency spectrum of the audio signal. It indicates where the bulk of the energy of the spectrum is concentrated. In this code, we compute the spectral centroid using the librosa.feature.spectral\_centroid function.
* **Spectral Rolloff**: This is a measure of the frequency below which a specified percentage (usually 85%) of the total spectral energy lies. In this code, we compute the spectral rolloff using the librosa.feature.spectral\_rolloff function.

**GTZANDataset Class:**

* In the \_\_**init**\_\_ method, data and labels are saved as instance variables.
* The \_\_**len**\_\_ method returns the length of the dataset which is the length of the data array.
* The \_\_**getitem**\_\_ method is responsible for returning a single sample from the dataset given an index.
* In this method, the data and labels are retrieved using the index and are converted into PyTorch tensors using torch.Tensor().
* The data tensor x and the label tensor y are then returned as a tuple.

**MyModel Neural Network:**

* This code block shows the modification of the architecture of a neural network. The original architecture was defined with 3 fully connected layers (also known as linear layers) with 22 input features, 128 hidden units in the first layer, 64 hidden units in the second layer, and 10 output units corresponding to the number of classes in the dataset.
* To increase the accuracy of the model, the number of hidden units in the first two layers have been increased. Specifically, the first layer has been modified to have 256 hidden units, and the second layer has been modified to have 128 hidden units. This change will allow the model to learn more complex patterns in the data and potentially improve its performance. The last layer remains the same as before, with 10 output units.
* It's worth noting that modifying the architecture is just one approach to improving model accuracy. Other approaches include changing the learning rate, adjusting the loss function, adding regularization, and increasing the number of training epochs, among others.



Chart, line chart

Description automatically generated

**Requriements:**

* pip install librosa