# Classy Muse: Music Classification Using Audio Analysis and Descriptive Metrics

Team members: Robert Turnage, Grant Wilson, Prachi Varma

## Problem Statement

Traditionally, music streaming services, such as Spotify, are provided musical genre metadata by labels and publishers. While this is generally reliable, it is not uncommon for metadata from smaller/indie publishers to be incorrect, leading to a bad user experience. Additionally, songs can often be categorized in multiple genres. This allows them to have a farther reach when run through recommendation and personalization algorithms.

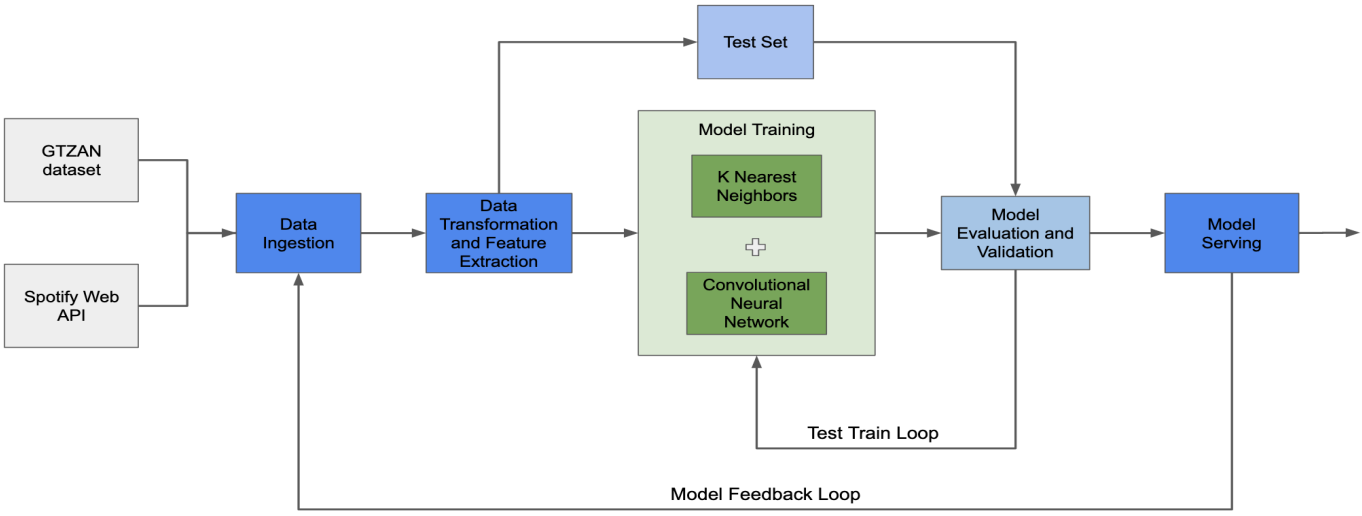
## Objective

We want to build an ML model that will classify different musical genres by audio analysis. Defining the many community described music genres we will attempt to classify by analyzing audio features of frequency and time alongside metrics of track “energy”, tempo, and speed.

## Approach/ Methodology

Features such as frequency patterns, for example salsa drum beat, country steel guitar reveals a potential genre given those specific categories of cultural influences in music. For our experiments specifically we want to focus on attributes which are important in audio files and music specifically. We are particularly interested in frequency, wavelength, amplitude, and pitch to see if we can identify differentiating factors in different genres. The overall process is described below in our block diagram.

# Block Diagram




First, we will get audio files and features from the GTZAN dataset and from spotify. Next, we will ingest the data and attempt to extract meaningful features for our models. The data will then be split into training and validation sets. Finally, We will run our models, evaluate the results and fine tune the parameters in the hopes of improving accuracy scores. This process will take a few rounds of iteration.

# Datasets

GTZAN genre classification dataset, can be constructed from personal audio library with possible need for similar sizes and frequencies. Spotify Web API data can pull audio features from Spotify tracks. We also were able to pull snippets of Spotify tracks from Everynoise.com in order to have access to a wider array of genres and tracks.

GTZAN genre classification dataset
http://marsyas.info/downloads/datasets.html

Spotify Web API
https://developer.spotify.com/documentation/web-api/

# What is considered success/ failure?

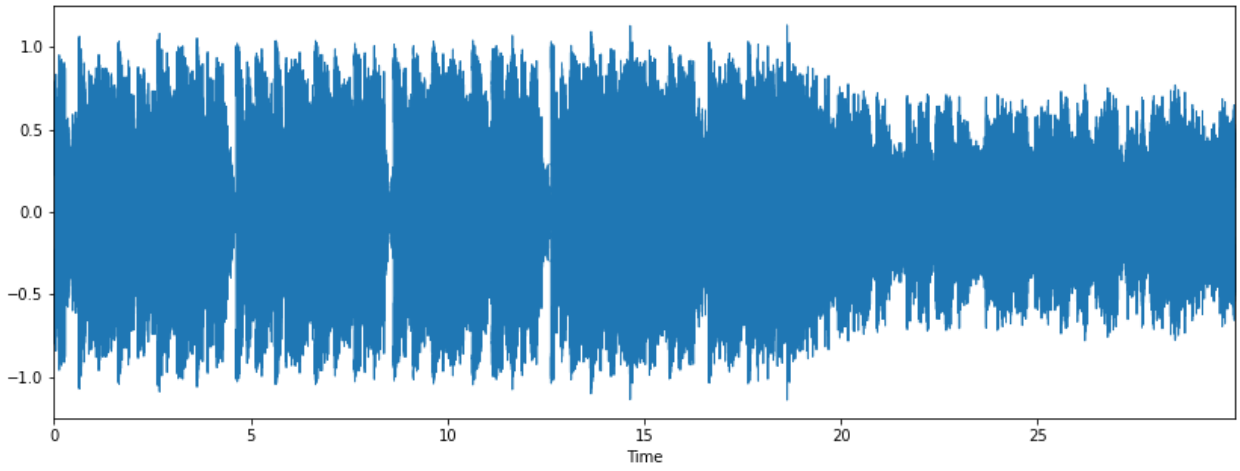
High degree of accuracy (>75%) when predicting a track's genre based on its features. Sub-genre or genre level.

# EDA and Feature Extraction

In order to run experiments by audio analysis we need to conduct quite a bit of feature extraction. We used a library called "librosa" to help with this analysis.

## Waveform and Zero-crossing Rate

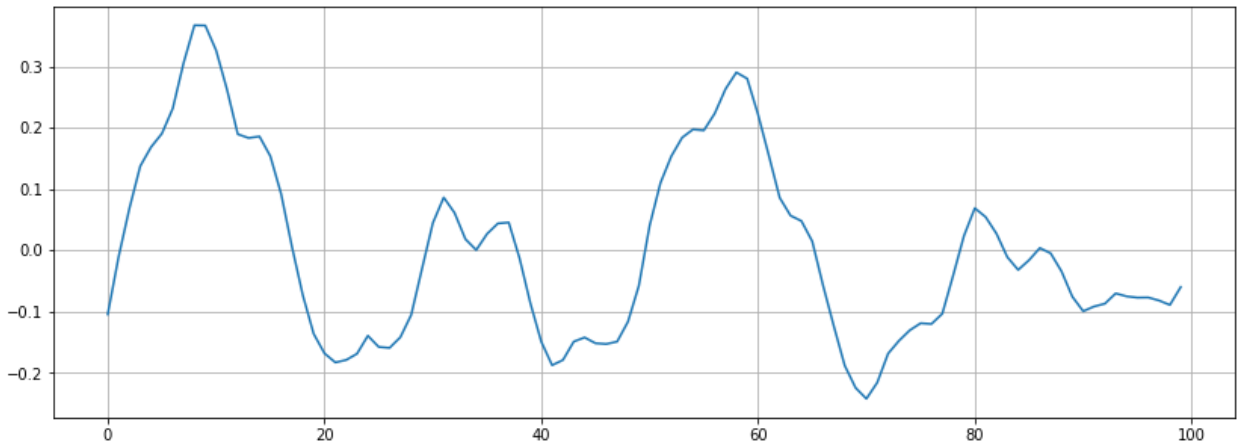
We started our EDA off by looking at the simple waveform of a 30 second clip of a song from the "dance pop" genre. A waveform is the visualization of a sound wave that graphically depicts the features of amplitude, frequency and wavelength.[1] The zero-crossing rate is the rate of sign-changes along with a signal, i.e., the rate at which the signal changes from positive to negative or back. This feature has been used heavily in both speech recognition and music information retrieval. It usually has higher values for highly percussive sounds like those in metal and rock.[2]




From the above image we can see that this song seems to have a high frequency and amplitude with shorter wavelengths. These characteristics are typical of dance pop songs and other high energy genres.

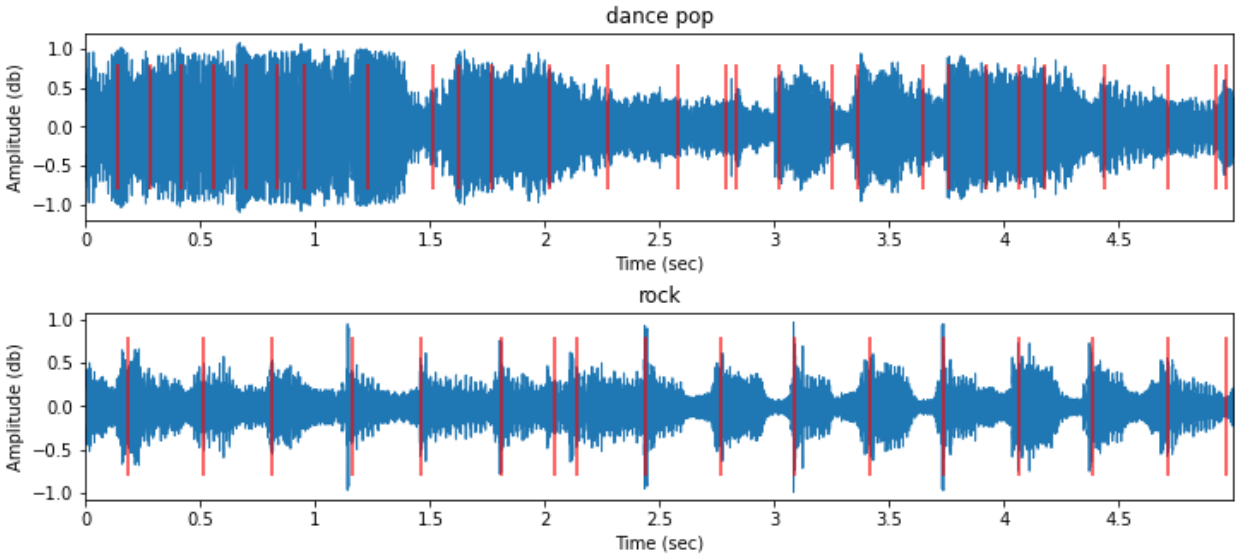
Let's zoom in a little closer to see what these wave patterns look like:

[1] https://www.soundassured.com/blogs/blog/what-are-sound-waveform-characteristics
[2] https://www.sciencedirect.com/topics/engineering/zero-crossing-rate




The above image shows a piece of the waveform present in the 12th second of this sample. This is about 18 milliseconds worth of audio, which contains multiple different frequencies and amplitudes. We can examine the zero-crossing rate in this visualization. These 18 milliseconds of this 30 second sample has a zero-crossing rate of 10.

Zooming back out we can compare the waveforms from a dance pop song to a rock song to see how the beat of a song can differ between these 2 genres.

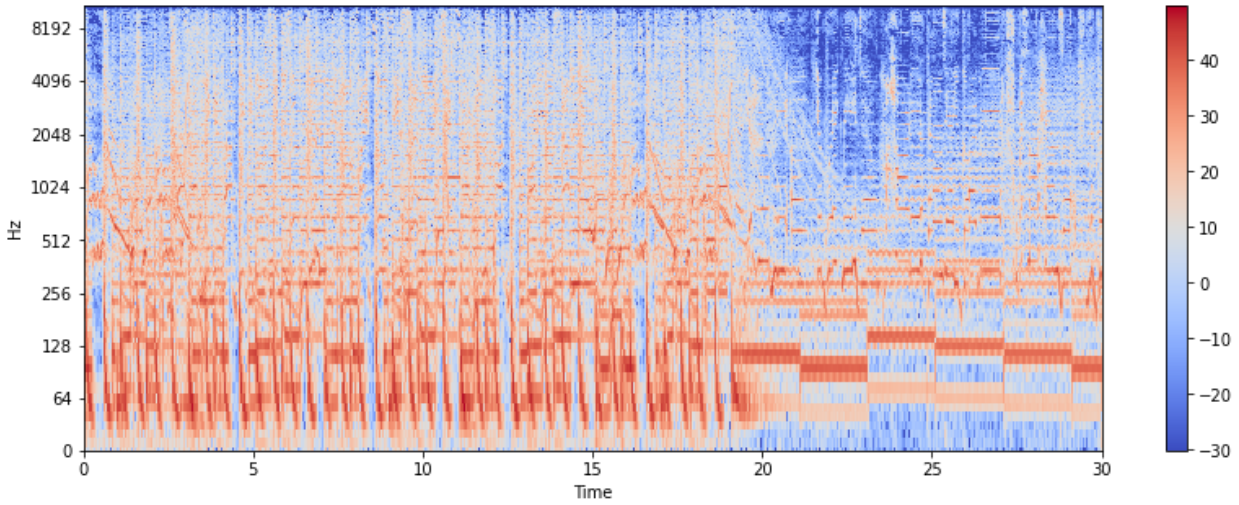



The red vertical lines denote the beat of the songs. In the dance pop song we see the beat is quite irregular or syncopated, meaning multiple rhythms are combined.[3] This has become quite common in this genre of music as it is often associated with danceability. Conversely, we can see the rock song contains one standard beat that flows through the entire tune of the song. Rock songs are often associated with more synchronized movements such as “head banging,” while putting more emphasis on lyrics and melody as the heart of the audio.

[3] https://iconcollective.edu/what-is-syncopation-in-music/

## Spectrograph

Next, we look at the spectrograph of the original dance pop song to get a better understanding of how the features are distributed in terms of amplitude specifically. A spectrograph allows us to visualize where the frequencies lie in the audio file.[4] The red areas indicate where the frequencies are concentrated while the blue areas contain more “dead” noise.




Note that we took the log of frequency to visualize the feature above as most of the “meat” was concentrated in lower hertz. This provides a nice visual representation of how this feature is distributed through this audio file.

## Spectral Centroids and Rolloffs

Spectral centroids indicate where the ” center of mass” for a sound is located and is calculated as the weighted mean of the frequencies present in the sound. Consider two songs, one from a blues genre and the other belonging to metal. Now, as compared to the blues genre song, which is the same throughout its length, the metal song has more frequencies towards the end, so the spectral centroid for blues song will lie somewhere near the middle of its spectrum while that for a metal song would be towards its end.[5]

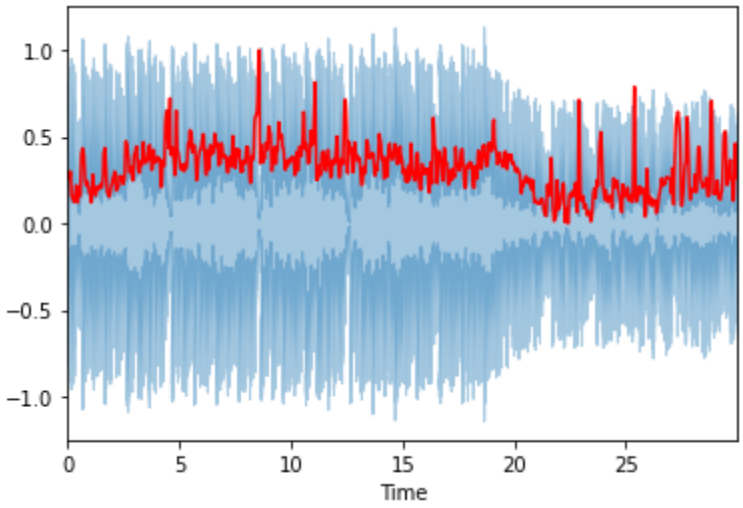
The roll-off frequency is defined as a measure of the shape of the signal. It represents the frequency below which a specified percentage of the total spectral energy, e.g., 85%, lies.[6]

Let’s look at our 30 second dance pop sample again.

First the spectral centroid:

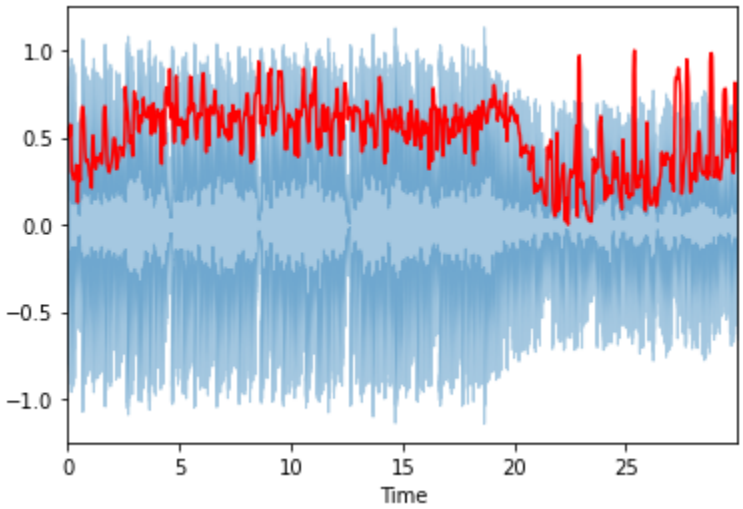
---

[4] https://towardsdatascience.com/learning-from-audio-spectrograms-37df29dba98c
[5] https://librosa.org/doc/main/generated/librosa.feature.spectral_centroid.html
[6] https://librosa.org/doc/main/generated/librosa.feature.spectral_rolloff.html




We can see for this audio the spectral centroid lies close to the middle, but mostly in the positive amplitude portion of the song.

The spectral roll off of dance pop song at 85% cut off:



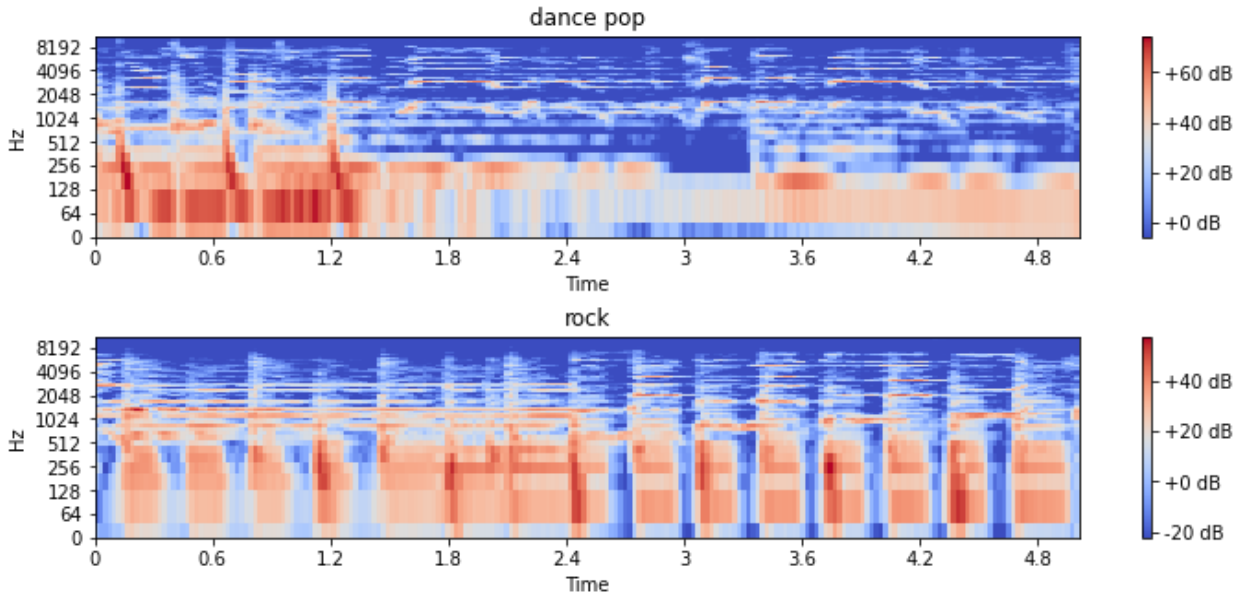

## Mel-Frequency Cepstral Coefficients and Chroma Frequencies

The Mel frequency cepstral coefficients (MFCCs) of a signal are a smaller set of features that concisely describe the overall shape of a spectral envelope. It allows us to focus on important features and cut out the “dead noise” that we saw in the spectrograph.[7]

The MFCC spectrographs of our dance pop audio vs a rock song:

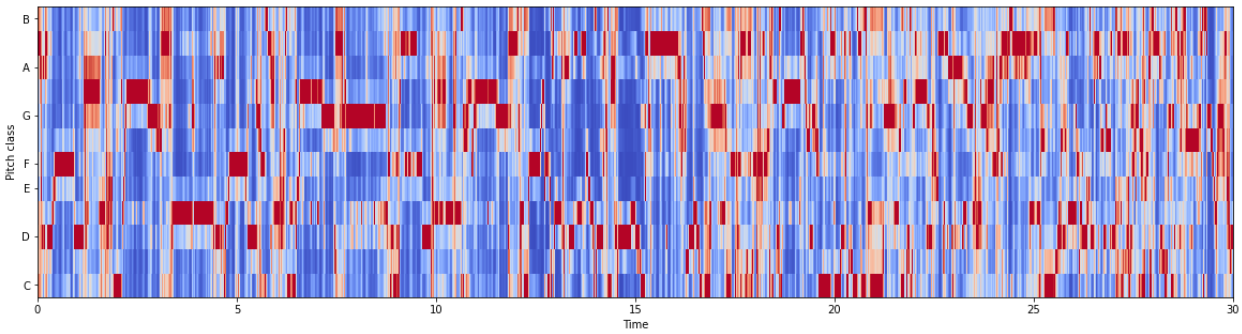
---

7 https://towardsdatascience.com/learning-from-audio-the-mel-scale-mel-spectrograms-and-mel-frequency-cepstral-coefficients-f5752b6324a8




Note that we took the log of frequency in this image inorder to intensify the area where features are concentrated. We can see here that we are mostly visualizing the “red” or important features that we saw in the spectrograph. We can see there are some differences in these two images specifically as the red areas are more evenly spaced out in the rock spectrogram versus the dance pop.

Additionally, chroma features are another representation for music audio. The spectrum is projected into bins representing the 12 distinct semitones of the musical octave.[8] This essentially creates a heat map of how the pitch changes over time.[9] Let’s see an example from our 30 second dance pop sample:




As we can see this song uses a range of notes particularly, A, G, D, C. This is a common chord progression in many popular songs, so it is not surprising that this is where notes are concentrated for this sample. However, they may not be specific to this genre, which could complicate training the model.

[8] https://www.ee.columbia.edu/~dpwe/resources/matlab/chroma-ansyn/
[9] https://towardsdatascience.com/learning-from-audio-pitch-and-chromagrams-5158028a505

# Experiments

For our models, we mainly focused on Convolutional Neural Networks and K-Nearest Neighbor models.

Why did we choose the CNN model option for Music Genre classification?

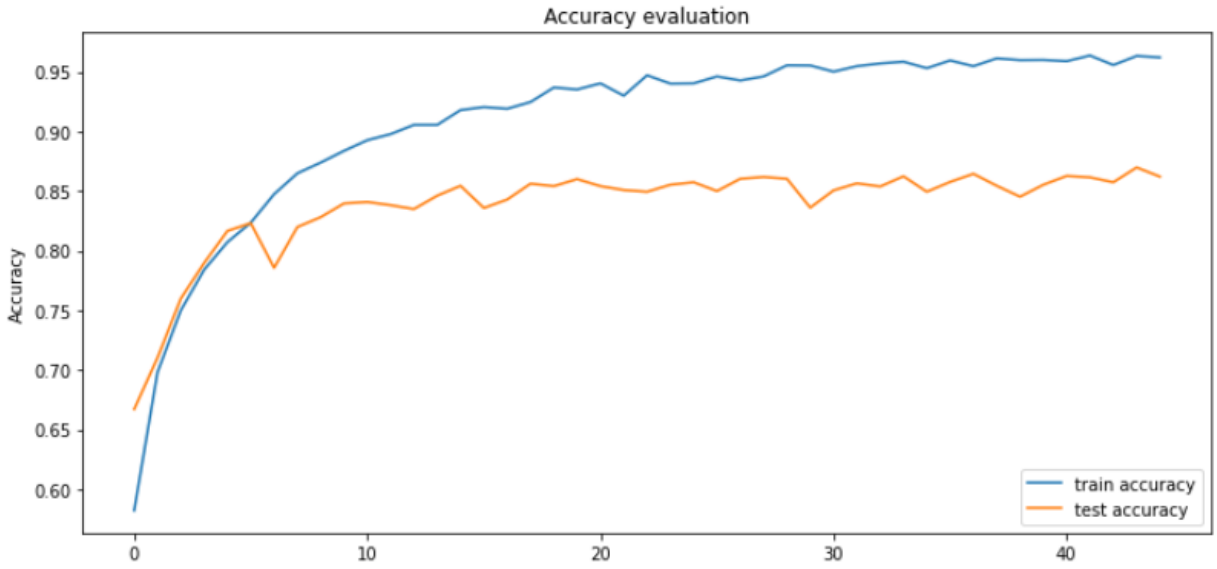
This was a matter of complexity in the data. Using librosa, if we extract the audio features through a MFCC Spectrogram, the richness of the data is very complex. In this process we have a visual representation of audio data. We also have the requirement for multi classification, and these two categories refine our options to usable from most obvious by design to least but possible last. We ruled out SVM and K-Means Clustering first for the following reasons. Support vector machine models can be used for regression or classification and as a general algorithm wouldn't expect as much accuracy. K-means clustering is an unsupervised learning method that wouldn't use the label data. We do have audio

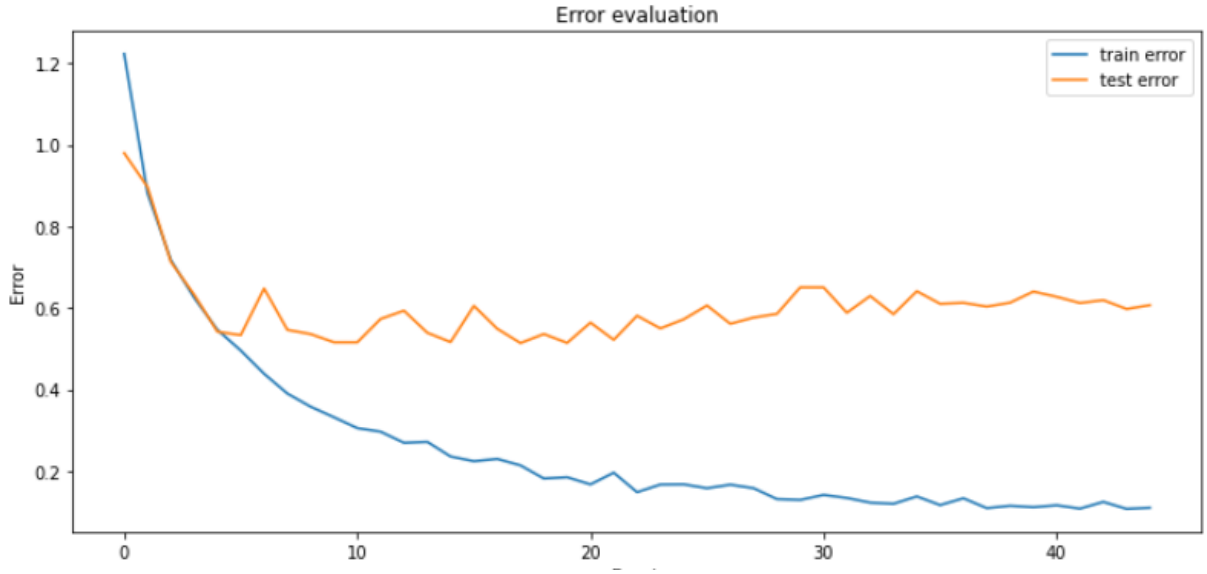
## CNN

This was the first model that came to mind after our data exploration and feature extraction. CNN is designed for visual recognition and the ingest data is visual data as we were able to extract data through the spectrographs above. Additionally, this model has the most tuning for visual classification and is capable of multi-classification. These experiments were run on both the spotify and GTZAN datasets.

```
Model: "sequential_21"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Conv2d_1 (Conv2D)           (None, 128, 11, 32)       320

 MaxPooling2D_1 (MaxPooling2  (None, 64, 6, 32)        0
 D)

 BatchNormalization_1 (Batch  (None, 64, 6, 32)        128
 Normalization)

 Dropout_1 (Dropout)         (None, 64, 6, 32)         0

 Conv2d_2 (Conv2D)           (None, 62, 4, 32)         9248

 MaxPooling2D_2 (MaxPooling2  (None, 31, 2, 32)        0
 D)

 BatchNormalization_2 (Batch  (None, 31, 2, 32)        128
 Normalization)

 Dropout_2 (Dropout)         (None, 31, 2, 32)         0

 Conv2d_3 (Conv2D)           (None, 30, 1, 64)         8256

 MaxPooling2D_3 (MaxPooling2  (None, 15, 1, 64)        0
 D)

 BatchNormalization_3 (Batch  (None, 15, 1, 64)        256
 Normalization)

 Dropout_3 (Dropout)         (None, 15, 1, 64)         0

 Flatten (Flatten)           (None, 960)               0

 Dense_1 (Dense)             (None, 128)               123008

 Dropout_4 (Dropout)         (None, 128)               0

 Dense_2 (Dense)             (None, 10)                1290

=================================================================
Total params: 142,634
Trainable params: 142,378
Non-trainable params: 256
_________________________________________________________________
```

After training 45 epochs, we found the following training and validation accuracies:




The training accuracy reached a peak of about 0.96, while the validation accuracy reached a peak of about 0.86. From these results, we can tell our model fits well at epoch 10 for an overall model evaluation of 87% accuracy.[10]

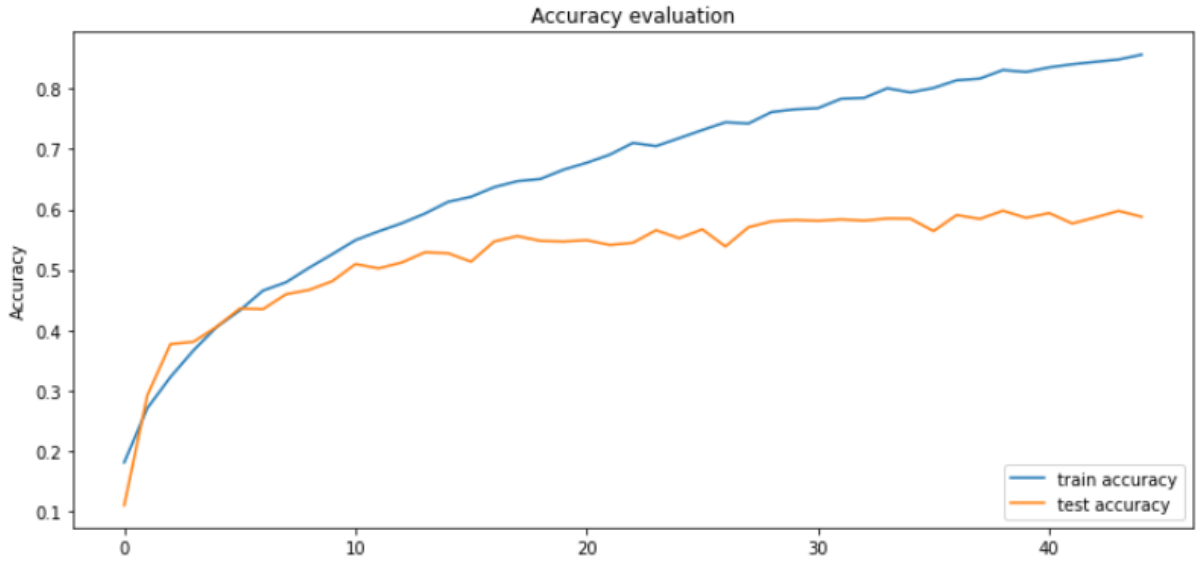
10 https://medium.com/@sidathasiri/building-a-convolutional-neural-network-for-image-classification-with-tensorflow-f1f2f56bd83b

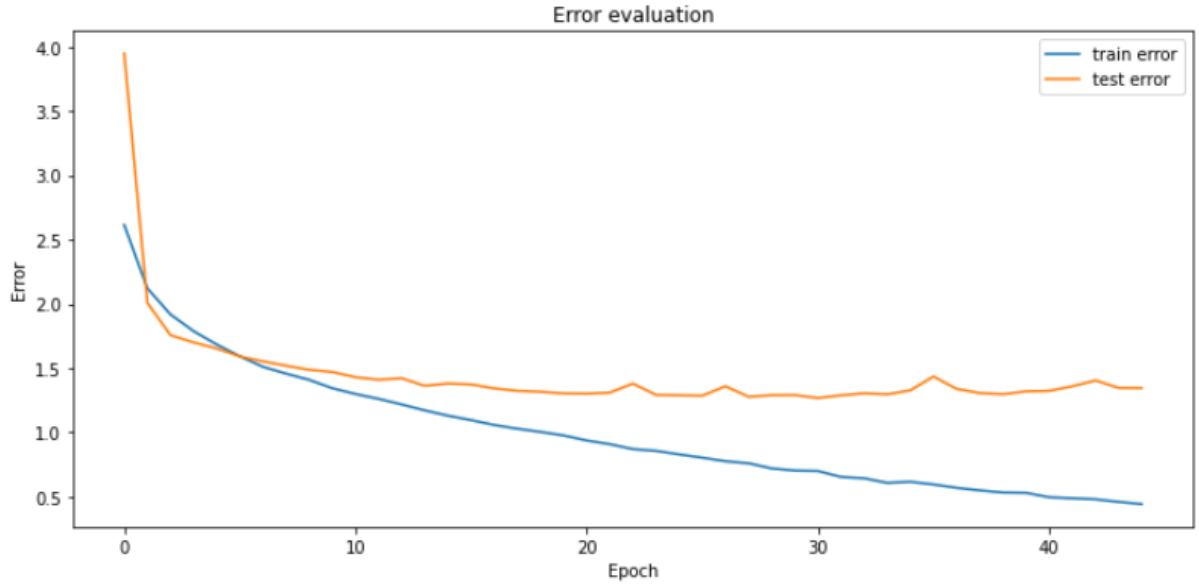
### CNN with Chroma features

Next, we wanted to see if we could get more accurate results using chroma features specifying pitch compared to the MFCC images above. The data was preprocessed using librosa to extract the chroma features from the spotify audio files. The model summary below:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 128, 11, 32)       320

 max_pooling2d (MaxPooling2D  (None, 64, 6, 32)        0
 )

 batch_normalization (BatchN  (None, 64, 6, 32)        128
 ormalization)

 conv2d_1 (Conv2D)           (None, 62, 4, 32)         9248

 max_pooling2d_1 (MaxPooling  (None, 31, 2, 32)        0
 2D)

 batch_normalization_1 (Batc  (None, 31, 2, 32)        128
 hNormalization)

 conv2d_2 (Conv2D)           (None, 30, 1, 64)         8256

 max_pooling2d_2 (MaxPooling  (None, 15, 1, 64)        0
 2D)

 batch_normalization_2 (Batc  (None, 15, 1, 64)        256
 hNormalization)

 flatten (Flatten)           (None, 960)               0

 dense (Dense)               (None, 128)               123008

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 10)                1290

=================================================================
Total params: 142,634
Trainable params: 142,378
Non-trainable params: 256
_________________________________________________________________
```

After training 45 epochs, we found the following training and validation accuracies:




The training accuracy reached a peak of about 0.96, while the validation accuracy only reached a peak of about 0.86. It looks like our model is overfitted more for this audio feature as compared to the mfcc.

### CNN + KNN Hybrid Model

K Nearest Neighbors is a supervised learning that is usually paired with OpenCV for visual recognition.Because KNN groups observations based on nearby values, it is the perfect tool for looking at metadata for help with classification and could work well as a hybrid model with CNN. Since the Spotify data is metadata for our track analysis, we used CNN to inform the KNN model and increase the accuracy and quality together.[11]

Our Hybrid model did well on our genre-level classification The training accuracy was 0.87, with F1-score's reaching 0.95. It looks like the model has benefit as a hybrid for an increase of overall accuracy of 1%.

## Limitations of the Study

The nature of music is constantly changing, including what we recognize as country, hip hop, or rock. These borders are not only vague, they aren't consistent; 90s hip hop and 00s hip hop are vastly different. This means our data, which is all modern, will apply mostly to the current musical landscape and may struggle with other decades if our model is not properly trained. Our data is also limited to 10 genres which ignores some of the finer details that differentiate sub-genres of music. For example, hip-hop of today differs significantly from the hip-hop of the 80s or 90s and our data treats them as one.

Additionally, we used 30 second audio snippets to create our audio features. This may be sufficient for genre classification, however there is much more to a song than the first 30 seconds which means we may be leaving useful information on the table that may help in further analyses.

## Future Work

The ability to classify a track based on it's audio features has plenty of application on its own. However, this work has broader implications for both genre classification and music recommender systems.

Current recommender systems, especially for music, often rely on user-based recommendations that rely heavily on the listening patterns of other users. While this is helpful for systems with vast collections of data and many users, it can be restrictive in systems with few users and for tracks that aren't popular since discovery requires listening activity by many other users. In effect, this restricts recommendations to the most popular songs listened to by the most users.

By focusing on an analysis of audio rather than listening patterns, our algorithm paves the way for feature-based recommender systems that can create recommendations based on interpreted genre rather than label-reported genre. These feature-based recommendations also

[11] https://www.analyticssteps.com/blogs/music-genre-classification-using-machine-learning

eliminate the data size issues with the user-based recommendation system. Additionally, feature-based recommendations will allow users to opt for new music and emphasize discovery over popularity. We may even be able to continue the analysis from genre-based recommendations into mood-based recommendations that would create playlists of music based on perceived mood and energy of a track.

Improvement to the current model could be increased if we preprocess the audio data to separate stereo channels. For each channel we can further separate vocals from instrumentals each having its own track.

## Standards

We used python version 3.6.9 for our project. Additionally, the following libraries were used for our experiments and results:

- `librosa`
- `sklearn`
- `scipy`
- `numpy`
- `pandas`
- `matplotlib`
- `spotipy`
- `ffmpeg`
- `tensorflow`