Genre Classification using Feedforward Neural Networks

Alexis Adzich, Andreas Koni, Natasha Mubeen

Introduction

With over 500 million listeners, Spotify is one of the largest music streaming platforms globally. Many users listen to their favorite songs and value the service for its personalized recommendations that are based on listener activity. Different factors influence these recommendations, from the songs and artists users listen to, to more detailed statistics like individual listening times for songs and which ones are skipped or liked. One of the broadest ways to differentiate individuals' music tastes and preferences is through genres of music, making an algorithm to classify the genre of songs very useful in several contexts. These genre classification algorithms play a crucial role in shaping the way users discover, consume, and interact with music and multimedia content in diverse applications and industries. Their ability to automatically classify and organize music content based on genre enriches user experiences, enables personalized recommendations, and drives innovation in music technology and entertainment.

Objectives

The objective of this project is to develop a genre classification model for songs using feedforward neural networks, leveraging the power of deep learning to accurately classify music tracks into predefined genres. Spotify publishes detailed data for each of its songs, measuring songs' 'danceability', 'valence', and 'speechiness', along with several other traits. We will train our neural network model on a large and diverse dataset of music tracks, analyzing each of the song's characteristics to determine what genre it most likely belongs to given its respective scores.

Motivations & Real-Life Applications

This genre classification algorithm is interesting because of its extensive list of real-life applications. Aside from eradicating the need for a person to manually listen to, judge, and label the genre of each song (which comes with its own biases), an automatic genre classification algorithm can be used as a valuable tool for several other purposes. Some examples include:

1. Music professionals can use the algorithm to gain insight into music genres and their characteristic traits to analyze trends, understand audience preferences, and gain further insight into the popularity of different music genres. This information can inform marketing strategies, content creation, and business decisions in the music industry.
2. A recommendation algorithm can use the results of the genre classification to recommend songs or curate playlists including songs from genres a user specifically likes.
3. Content creators, broadcasters, and rights holders use genre information to license and distribute music content across different platforms and media channels. Genre

classification algorithms may assist in music licensing and rights management by categorizing music tracks based on their genres and usage rights.

There are a few other approaches that could have been used:

- We initially planned to implement a KNN or GMM clustering algorithm, where we clustered our entire dataset based on all metrics and then used the most common genre of each cluster as the predicted genre for songs belonging to the cluster. However, we were not getting high accuracies and so we looked towards neural networks.
- We could have also used spectrograms or other time-frequency representations of songs for them to be treated as images and then used Convolutional Neural Networks to learn hierarchical features from these representations for genre classification.
- We could have also used Recurrent Neural Networks, particularly Gated Recurrent Units (GRUs) as they can capture temporal dependencies and long-range patterns in sequential audio data, making them suitable for tasks such as music genre classification where the order of audio features may be important.

However, we ultimately decided on dense connected layers for genre classification due to its simplicity and interpretability as well as the fact that we had enough available computational resources for it.

Description of Data

We obtained our dataset of songs through Kaggle.
This dataset contains 1000 songs from 114 genres of music. Each of the songs has the following metrics (metric descriptions taken from the source data):

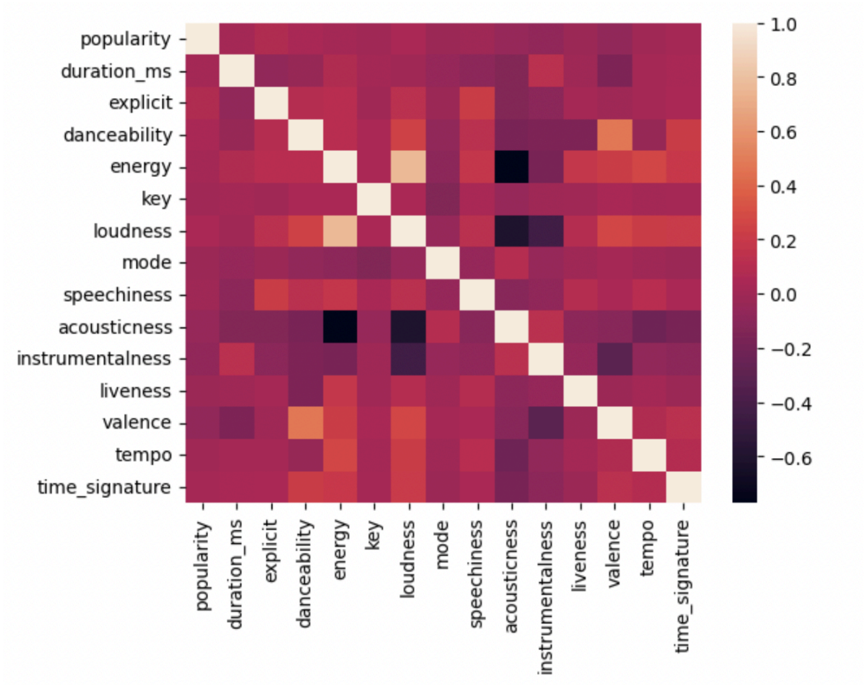
Feature	**Description**	**Type**
artists	The name(s) of the artist(s) associated with the song.	Str
album_name	The name of the album that the song belongs to.	Str
track_name	The name of the song.	Str
popularity	The popularity score of the song on Spotify, ranging from 0 to 100.	Int
duration_ms	The duration of the song in milliseconds.	Int
explicit	A boolean value indicating whether the song contains explicit content.	Bool
danceability	Score representing how ‘danceable’ a song is based on its musical elements, ranging from 0 to 1	Float
energy	The score measuring the intensity and activity of a song, ranging from 0 to 1.	Float
key	The key of the song, represented by an integer value.	Int
loudness	The loudness of the song in decibels (dB).	Float
mode	The tonal mode of the song, represented by an integer value (0 for minor, 1 for major).	Int

speechiness	A score from 0 to 1 that represents the presence of spoken words in a song.	Float
acousticness	A score from 0 to 1 that represents the extent to which a song possesses an acoustic quality.	Float
instrumentalness	A score from 0 to 1 representing the likelihood of a song being instrumental.	Float
liveness	A score from 0 to 1 that represents the presence of an audience during the recording or performance of a song.	Float
valence	A score from 0 to 1 that represents the musical positiveness conveyed by a song.	Float
tempo	The tempo of the song in beats per minute (BPM).	Float
time_signature	The number of beats within each bar of the song.	Int
track_genre	The genre of the track.	Str

Data Preprocessing

Our first step was actually inspecting the dataset and defining all the features. We saw that there were 114 genres with 1000 songs each, meaning that we had a decent sample size for each genre to train the model on and we did not need to drop any genres due to them having insufficient data. The fact that there were several genres also meant that our dataset was diverse and would give the model a wide range of data to train on. We then began processing the data by dropping all untrainable columns with string values including the track id, artist name, album name, etc., leaving us with just the quantitative metrics we were training on. To convert our categorical data, we first converted our boolean explicit feature into binary integers that our model can train with. We also had integer data for time_signature and key, that we converted into multiple one-hot encoded dummy variables so our model could train on each factor individually without influence from the intrinsic sequential hierarchy of the values.

However, we noticed that some of the metrics seemed similar conceptually- For example, we thought songs that might have a high valence (musical positiveness) might be more danceable, or songs that have more energy might have a higher liveliness score. We then needed to check whether certain metrics were highly correlated with one another as this could result in multicollinearity and therefore cause overfitting, or could increase model complexity and thus how computationally expensive our model would be. Therefore, we looked at the covariance matrix of our data to see if we needed to use PCA or other dimensionality reduction processes for our data.



Given our metrics were fairly uncorrelated, we decided not to drop any metrics. We then split our data into training and testing data with a test size of 0.2. Lastly, we used MinMax normalization to normalize each training set.

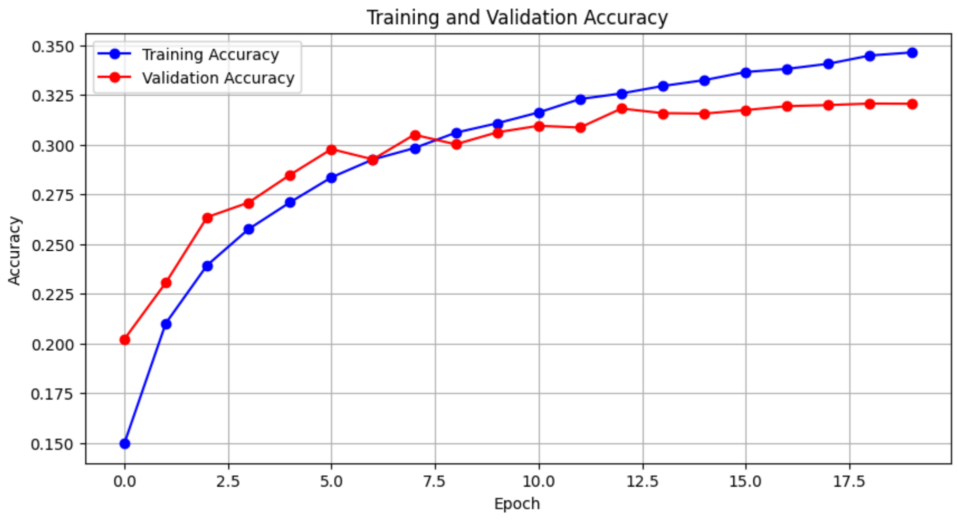
Initial Model

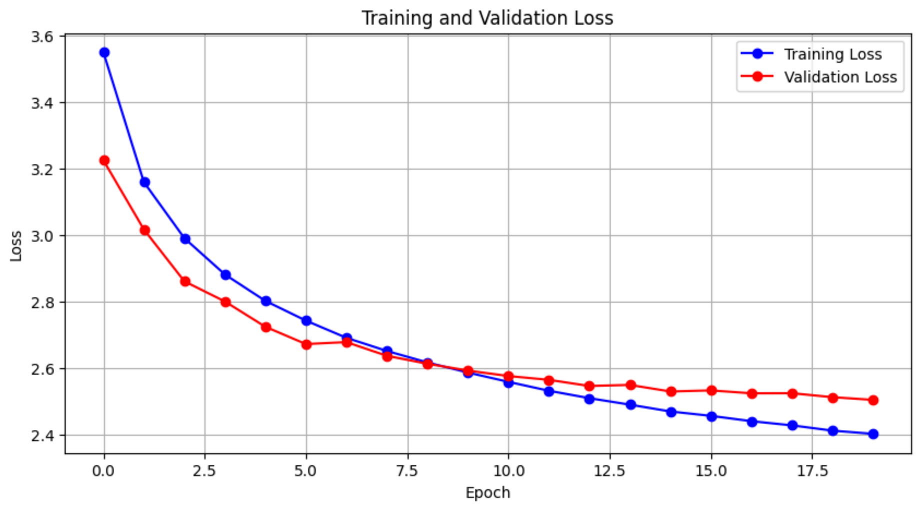
After splitting our data into a train and test set, we then started with creating a neural network to predict a track's genre based on its features. As discussed in our motivations, we decided to train our data on a fully connected neural network. After experimenting with different parameters and layers, we chose a model with two fully connected dense layers, followed by a batch normalization and dropout layer, and then two more dense layers to finally narrow down our model into 114 classifications. The model is as follows:

```
Model: "Model 1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 512)               15872

 dense_1 (Dense)             (None, 256)               131328

 batch_normalization (Batch  (None, 256)               1024
 Normalization)

 dropout (Dropout)           (None, 256)               0

 dense_2 (Dense)             (None, 128)               32896

 dense_3 (Dense)             (None, 114)               14706

=================================================================
Total params: 195826 (764.95 KB)
Trainable params: 195314 (762.95 KB)
Non-trainable params: 512 (2.00 KB)
_________________________________________________________________
```

Our model introduces some diversity in types of layers with fully connected dense layers, batch normalization, and dropout layers, which helps our model generalize to unseen data. We also introduce more moderate parameter sizes, for a total of just under 200,000, to ensure that our model is able to train on our large dataset, while also being conscious of high complexity in overfitting concerns.

While training our data, we researched methods to help improve our final accuracy score. One of these was to add a learning rate scheduler that gradually decreases the learning rate at every epoch, in our case by a factor of 0.9, which allows our model to make larger learning steps in the beginning, and then make more fine-tuned changes as each epoch completes and it is able to pick up more specific features between our more nuanced genre differentiations.





After training the model, we achieved about 36.80% accuracy and 33.39% validation accuracy. While this is a rather low accuracy percentage, the sheer amount of genres to classify likely is one of the biggest limitations our model faces. We explored some of these categories' precision and recall scores for the fitted test data to identify which genres the model struggled to classify.

Many of these genres included languages such as *indian*, or genres such as *j-idol*, that may resemble similarity to *j-pop* and *j-dance*, decreasing the model's ability to make these precise predictions.

Clustering

To improve the accuracy of our model, our first intuition was to narrow down the number of genres since trying to predict 114 different genres felt like the most obvious limitation on our model. The various genres included many ambiguous ones, as well as ones that seemed extremely similar or basically identical, such as *reggae* and *reggaeton*. So, our first approach was to use clustering algorithms like KMeans, Spectral Clustering, and Affinity Propagation.

For both KMeans and Spectral Clustering we had to specify the number of clusters we desired the algorithm to narrow the genres down to. We wanted to greatly decrease the number of genres from 114, so we decided on 20 clusters. Before clustering the genres, we first had to find the averages of each trait for each genre. So for the pop genre, for example, we took the average of the 1000 pop songs for each trait and set that as the value for the pop genre. We repeated this for the rest of the genres, leaving us with a data frame with 114 rows and 15 columns. Next, we standardized the data by using the *MinMaxScaler()* function from the *sklearn.preprocessing* package in order to reduce the computation struggle when clustering while also retaining as much information as possible about the shape and distribution of the data. Now, from the *sklearn.cluster* package, we ran KMeans and Spectral Clustering with 20 clusters to group the genres.

There were certainly some notable similarities and differences between the clusters that the algorithms generated. For example, KMeans had grouped *detroit-techno*, *minimal-techno*, *chicago-house*, *trance*, and *techno* together in a cluster. Interestingly, Spectral Clustering grouped *minimal-techno*, *trance,* and *techno* in the same cluster and grouped *detroit-techno* and *chicago-house* in another cluster. It is informative to see that these genres do have a pretty clear similarity between their musical traits, but also that perhaps Spectral Clustering was able to differentiate even further between the genres. In a broader sense, it reinforces the idea that clustering the genres could potentially contribute to the improvement of our model.

We also wanted to try Affinity Propagation as another clustering technique because it did not require a user input for the number of clusters. We thought that maybe specifying the number of clusters was limiting the efficiency of clustering the genres. Hence, Affinity Propagation provides the flexibility in the number of clusters. Once again, we use Affinity Propagation through the *sklearn.cluster* package and run it on our averaged and then standardized data. The algorithm gave an output of 15 clusters, which brought to question whether it would have an effect on the accuracy of our model, which we will see later on.

One clear feature we noticed with all three clustering algorithms was the discrepancy between the sizes of the clusters. For example, the genre *comedy* was always in a cluster of its own while there also being several clusters with 10+ genres in them. This unbalanced nature of the sizes of the clusters could affect the prediction of our models because the significantly larger clusters could overwhelm the other singular clusters. Thus, to try to combat this bias, for each algorithm, we took a random sample of 1000 from each cluster that it generated and created a new dataset to train on our model. This took advantage of narrowing the genres down to a smaller set while also preventing any possible bias in our predictions.

As a final clustering technique, we clustered the genres by our intuition and understanding of subgenres and genres alone, without any aid from clustering algorithms. This process included grouping similar genres and also dropping genres we felt were too ambiguous, such as *British* or *comedy*. This technique could act somewhat as a control for us to compare with the clustering algorithms. Next, we train our model on each of the clustering techniques we used and see which ones gave the best results in terms of predicting genres.

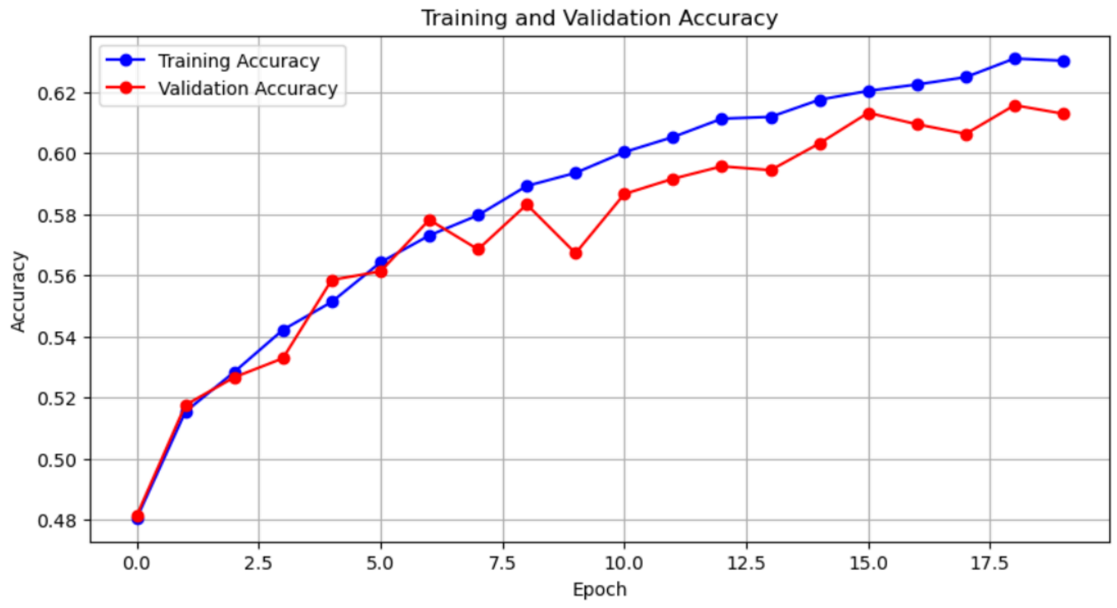
Training on Clustered Data
To evaluate our various clustering algorithms, we applied the model to our relabeled data, replacing our genre labels with labels corresponding to its respective genre cluster. We chose to train all of our datasets on the same model described previously, with a batch size of 32, a learning rate scheduler, and for 20 epochs.

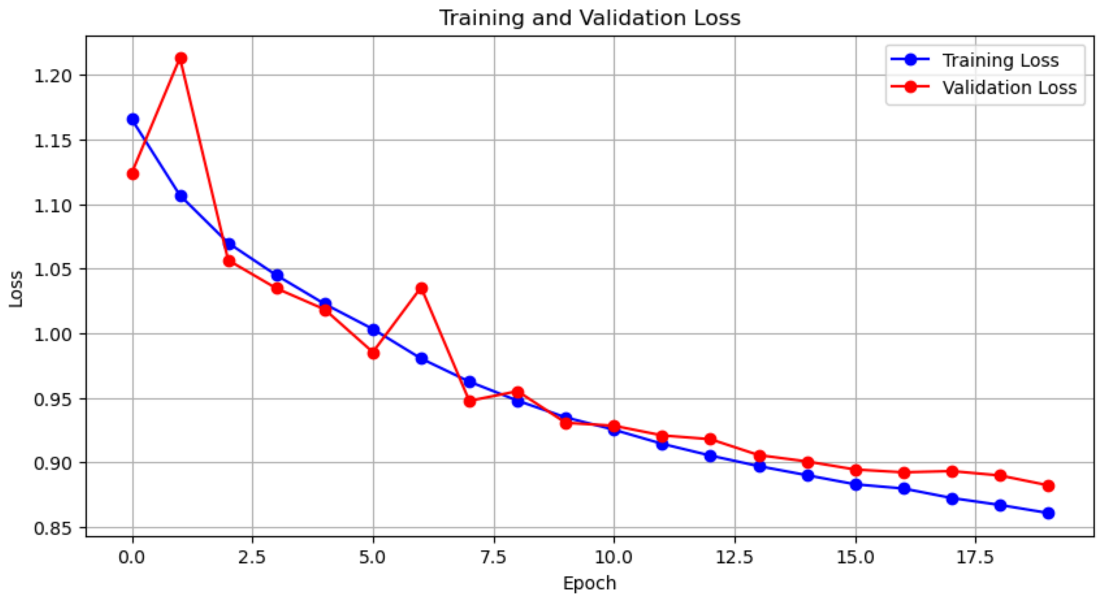
Our final results from each method are as follows:

Clustering Method	**Number of Clusters**	**Final Validation Accuracy**
KMeans	20	0.5356
Spectral	20	0.5669
Affinity Propagation	16	0.5367
Manual Clustering	5	0.6128

Final Model Evaluation and Analysis
Amongst the sklearn packages, the model performed rather similarly across clustering algorithms, achieving the highest validation accuracy with spectral clustering with 56.69% accuracy with 20 clusters. Our own clustering algorithm outperformed the other clustering methods with a test accuracy of 61.28%. We can likely attribute part of this increase to our choice in genre reduction, dropping ambiguous categories such as *german* or *sleep*, the model may struggle to distinguish. Some interesting discoveries we found evaluating our final model

predictions were the types of genres our model more consistently predicted or struggled with. We found that our model often achieved high levels of precision and recall in "rock" type genres, and often struggled more with identifying genres within "pop" clusters. This might be due to more variability in what's defined as pop music, or that rock has a more distinct sound that can be differentiated from other genres.





Above, we plotted our training and validation accuracy and loss over epochs to identify where our model starts to plateau and also to identify signs of overfitting. Our accuracies remain fairly close as epochs increase, and while we do see some difference in a higher training accuracy compared to our validation, we do not see significant evidence of concern for overfitting. And as we approach 20 epochs, we do see the curves flattening out to stabilize at around 62% accuracy and a loss of 0.875.

Model limitations/ Future questions

1. *How are songs in other languages classified differently?* The dataset had genres such as "British", "French", "Brazil ", "German", "Malay", "Swedish", etc. These are clearly classified based on the language/ country of origin of the song rather than the genre of the song itself- For example, the genre "British" makes British pop songs indistinguishable from British rock songs when running our model. This might make it difficult for the model to accurately classify them based on genre. One could further investigate how the songs within the genre would be classified amongst the other non-country genres- would there be the same number of songs from each non-country genre within the genre, or are certain genres overrepresented or underrepresented in the data, introducing bias?

2. *How does the lyrical content of the songs affect genre classification?* What if there were happy-sounding sounds that had sad lyrics, and were therefore considered to belong to the "sad" genre? Because our model does not take lyrical content into consideration, it might be misclassifying songs by only looking at their audio features. One could further investigate this by using natural language processing to judge the message and mood of lyrics in the tracks to improve our classification algorithm.

3. *How many genres should be clustered together when relabelling songs?* While our clustering algorithm significantly improved our accuracies, we noticed that some clusters had 5+ genres in the cluster, while some, like "Comedy", would just have themselves in its cluster. Our original data with 1000 songs from each genre ensured balanced sample sizes across the genres, which is important to avoid bias towards majority classes. However, if some clusters have a larger number of some original genres represented compared to others, randomly selecting 1000 songs from these n*1000 songs could lead to imbalanced representation within the clusters, and therefore the newly relabelled genres. Imbalanced representation may affect the model's ability to generalize well to all genres, particularly those with fewer samples within the genre.

4. *How should the model account for genre homogeneity?* In addition to there being genre imbalance within the newly labeled genres, some newly labeled genres may consist of songs that are more homogeneous in terms of musical style or characteristics, while others may exhibit greater diversity. The level of genre homogeneity within clusters can affect the model's ability to learn meaningful representations and boundaries between genres. One could further investigate how the extent of genre homogeneity affects the accuracy of the model.

5. *How could we evaluate model prediction proximity?* After performing our initial model to classify on all 114 genres, we relabeled our genres according to clustering algorithms to

reduce the hyperspecificity required for our model to train on so many nuanced genres. However, this grouping of genres, particularly if the cluster was large, may lead to high variance existing within genre clusters and hinder our model. One avenue to pursue is maintaining a full classification model on all 114 genres, and evaluating prediction accuracy utilizing our clusters to evaluate a metric of “closeness” if our model failed to predict the exact genre, but still succeeded in making a prediction within the same genre cluster.

Further extensions

Hybrid models: As mentioned before, there are several other approaches that could have been used, including the use of CNNs and RNNs. While we prioritized a dense fully connected neural network due to simplicity, this might have come at the cost of potentially lower accuracies. To extend this project, one could investigate the effectiveness of hybrid models that combine different neural network architectures, such as CNNs, RNNs, and fully connected layers. Leveraging the strengths of each architecture might mean that the hybrid models may achieve better performance in genre classification.

Data Augmentation: Songs are continually edited and covered (eg. Slowed and Reverbed remixes, Nightcore remixes, etc.), without it necessarily changing the genre. While our current model does not know how to deal with this, one could extend this project by augmenting the training data with synthetic samples generated through techniques like time stretching, pitch shifting, or adding background noise. The data augmentation can increase the diversity of the training dataset and improve the model's ability to generalize to unseen variations in the data, as well as improve its classification for remixes.

GitHub Link to Dataset and Code

https://github.com/aadzich/M156_Project

References

(2023 Dec). Spotify Tracks Genre, Retrieved 02/29 from https://www.kaggle.com/datasets/thedevastator/spotify-tracks-genre-dataset

(2021 Jan). Random Oversampling and Undersampling for Imbalanced Classification, https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/

(2007 Feb). Affinity Propagation Clustering Algorithm, https://scikit-learn.org/stable/auto_examples/cluster/plot_affinity_propagation.html