Andres Espinosa
Oronde Gibson
Leonardo Aramayo

ESI4610 Final Report: Song Genre Clustering and Classification

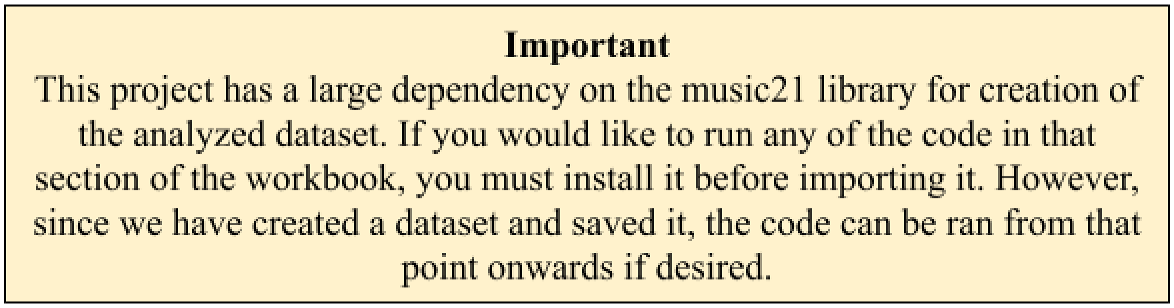
Table of Contents

Introduction

Our project is a machine learning multi-classification model that works with a dataset of 17,000 midi files (The Cleaned Lakh). We combined that dataset with another dataset of popular songs and artists and their genre to create a dataset with each song in the Lakh dataset and its respective genre. We then imported a music manipulation package called music21 and created our own music_processor.py file containing a series of functions and mappings that we then used to analyze a subset of 500 songs from the Cleaned Lakh. We created a dataframe with an addition of 18 columns with this analysis to each song. We then performed a clustering on the dataset to analyze if a Kmeans algorithm could provide insights to the data. We then ran a Random Forest classifier to predict the genre for each song from the dataset. Our conclusions led us to believe that, with the features extracted, the genres could not reliably and accurately be predicted.

This is a significant uncovering because it showed us a flaw in the general human classification system of genres. For example, the labeled genre for Nirvana is "Alternative" and "Blues" for Jimi Hendrix. However, most listeners will agree that those could also easily be lumped into the "Rock" category. Therefore, in order for us to accurately classify genres of songs we likely need to explore better options of classifying genres as humans.

A critical issue and challenge we encountered was processing time. The processing time for each song can range from 5-15 seconds depending on the size and if it has been processed before. This means that analyzing the full dataset can take over 70 hours of uninterrupted processing to complete. The more complicated the analysis, the higher the processing time so we had to cut a few analysis functions due to the time it took.



Important

This project has a large dependency on the music21 library for creation of the analyzed dataset. If you would like to run any of the code in that section of the workbook, you must install it before importing it. However, since we have created a dataset and saved it, the code can be ran from that point onwards if desired.

Objectives

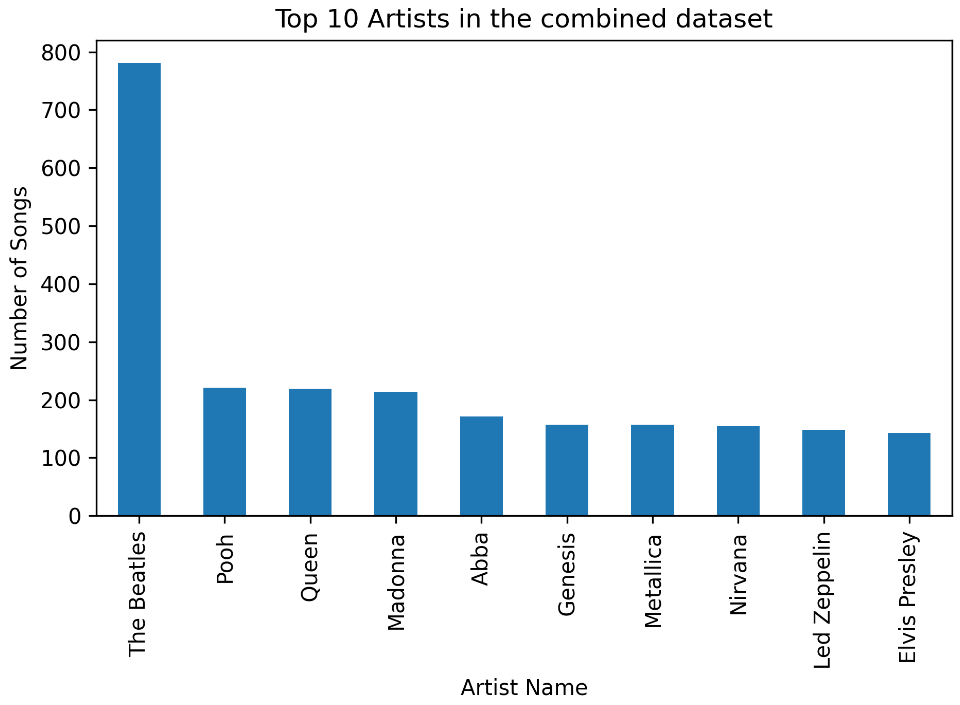
The rise of language learning models, exemplified by advancements such as ChatGPT, and the emergence of generative art models like DALL-E, have transformed the way humans can work with visual media. On the other hand, the domain of generative audio models has not yet experienced a comparable level of advancement, and it currently lags behind the already mentioned models.

The dataset our group decided to explore was the Cleaned Lakh Midi Dataset from Kaggle. It is a famous collection of midi files which are a special type of audio file that is much easier to manipulate than other file types such as wav or mp3. It is a collection of popular songs from artists like The Beatles, The Notorious BIG, and others. However, the dataset was given in the form of directories, such that each directory name was the artist/band name, and within them we could find the MIDI files named after a song.

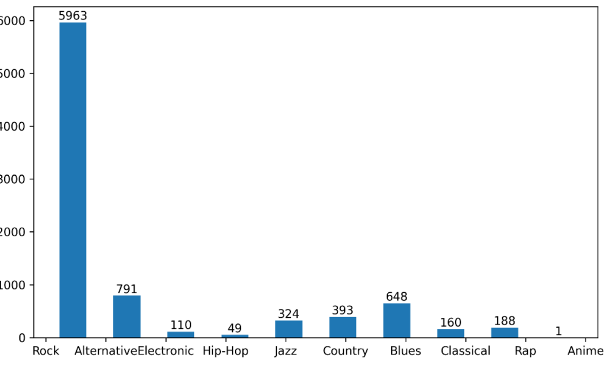
The primary objectives of our project were to construct a unified dataset containing the information of the songs and their respective MIDI files in order to develop a machine learning model that could generate songs or help us uncover necessary and critical features of songs that could serve as a foundation for future models with the same concept of generation through smart algorithms. By sharing our progress, we hope to encourage a broader community to explore this area where AI and human creativity are able to converge, redefining the landscape of musical composition in unprecedented ways.

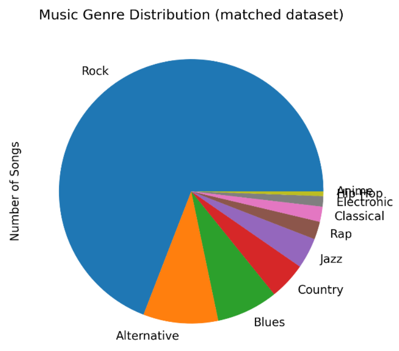
Methods

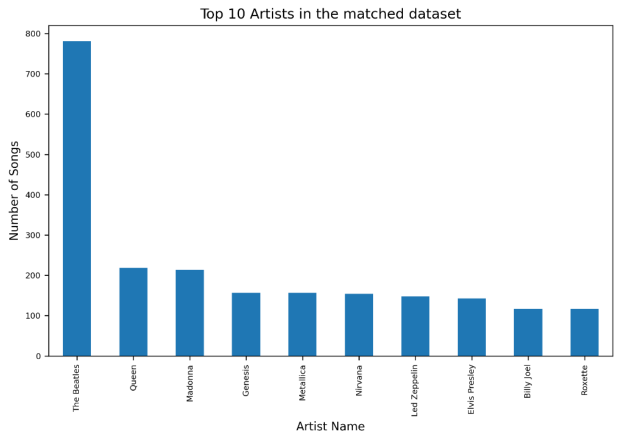
The first step in the project was loading all of the songs from the folders into one dataset. The structure of the folders is Cleaned Lakh/*Artist Name*/*Song file.mid.* In order to do this we used the **os** package to create a dataset with 3 columns: artist_name, title, and file_path, song_artist. To visualize the data, we created a bar graph with the frequency of each artist and their song.

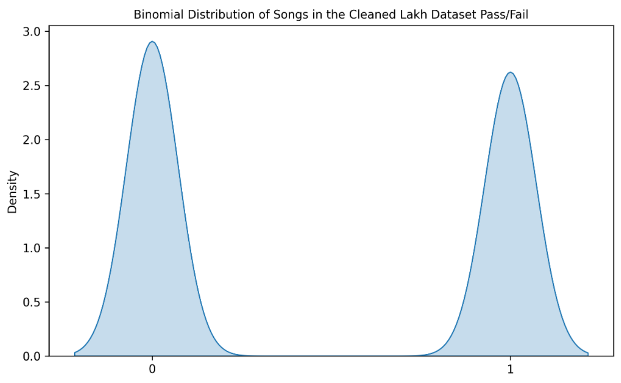


We then imported a dataset that had a list of songs and their genres to match the artist's top two genres to them in the main dataframe. We outputted a summary of the number of artists and their number of genres from the artists that we could match to the new dataset.

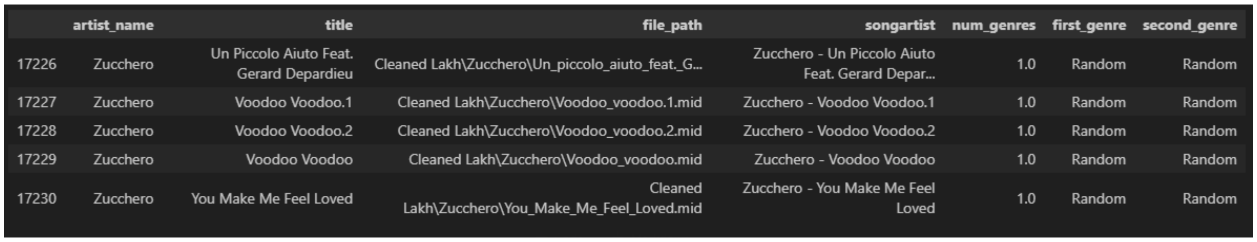








From matching this dataset to our Lakh dataset we were able to correctly classify about 52% of the artists. Despite the fact that we classified 52% of the artists, these artists comprised of about 12,000 of the 17,000 songs. This is due to the fact that a lot of the songs coming from smaller portion of artists so a higher portion of them were able to be classified. From this dataset, we added the "Random" genre to the rest of the songs that were not able to be classified so that we could still include them in the dataset for training. At this point, our Lakh dataset with genre added to it had this schema:

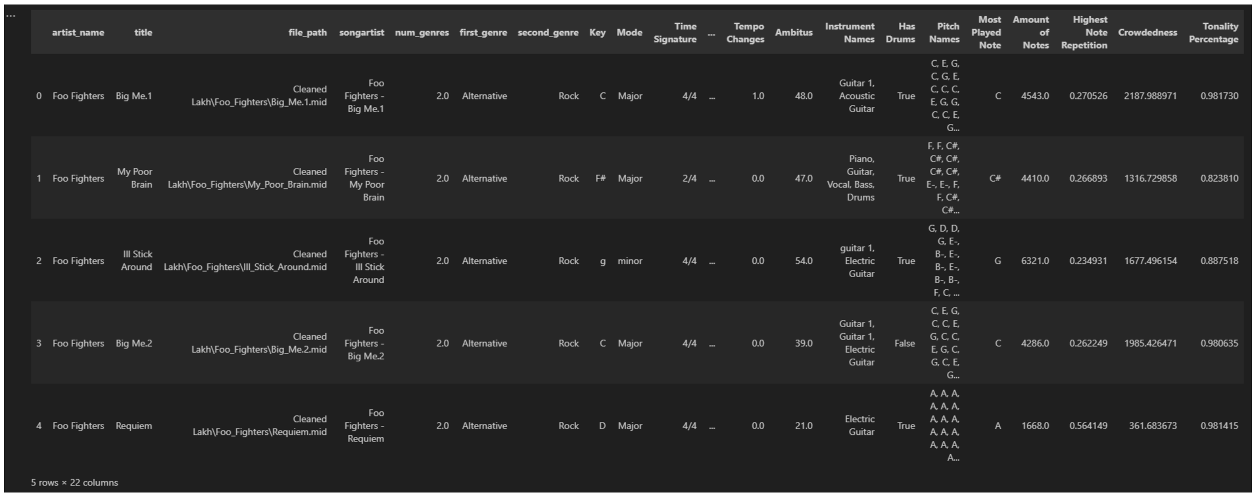


	artist_name	title	file_path	songartist	num_genres	first_genre	second_genre
17226	Zucchero	Un Piccolo Aiuto Feat. Gerard Depardieu	Cleaned Lakh\Zucchero\Un_piccolo_aiuto_feat._G...	Zucchero - Un Piccolo Aiuto Feat. Gerard Depar...	1.0	Random	Random
17227	Zucchero	Voodoo Voodoo.1	Cleaned Lakh\Zucchero\Voodoo_voodoo.1.mid	Zucchero - Voodoo Voodoo.1	1.0	Random	Random
17228	Zucchero	Voodoo Voodoo.2	Cleaned Lakh\Zucchero\Voodoo_voodoo.2.mid	Zucchero - Voodoo Voodoo.2	1.0	Random	Random
17229	Zucchero	Voodoo Voodoo	Cleaned Lakh\Zucchero\Voodoo_voodoo.mid	Zucchero - Voodoo Voodoo	1.0	Random	Random
17230	Zucchero	You Make Me Feel Loved	Cleaned Lakh\Zucchero\You_Make_Me_Feel_Loved.mid	Zucchero - You Make Me Feel Loved	1.0	Random	Random

With our labels attached to our dataset, we needed to find a way to extract features from the midi files and append those columns to our dataset. One issue we ran into is that our feature extraction is quite extensive and makes use of the music21 library created by MIT. Our solution to this was to factor out the code and create a music_processor.py file that consists purely of functions and mappings for use in an "analyze_songs" function. This function takes the midi file path, creates a music21 Score object from it, and then uses our custom music_processor.py file functions to analyze it. This part of the project took a large portion of the time of the project because we had to extract 18 features that otherwise did not exist from an audio file. Some of the features we extracted were:

- Key
- BPM
- Song Duration
- If the song has drums
- Crowdedness constant
- Tonality Percentage'

After creating this function, we created a list of 10 songs from 5 artists from each of the 10 genres and passed those to the analyze_songs function to add create individual dataframes for each genre. This function took around an hour for us to run so we outputted our features into a csv file titled "Analyzed Lakh Dataset.csv" so that we would not have to run that code every time we ran the workbook. The code is currently set to attempt to read the file from a csv, and if it fails, to create the dataframe from scratch. Below is a screenshot of this analyzed dataset.



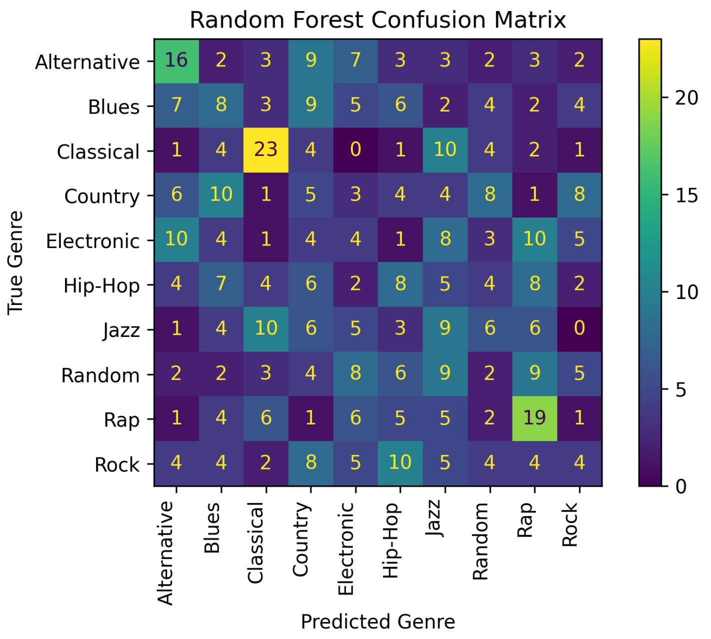
	artist_name	title	file_path	songartist	num_genres	first_genre	second_genre	Key	Mode	Time Signature	...	Tempo Changes	Ambitus	Instrument Names	Has Drums	Pitch Names	Most Played Note	Amount of Notes	Highest Note Repetition	Crowdedness	Tonality Percentage
0	Foo Fighters	Big Me.1	Cleaned Lakh\Foo_Fighters\Big_Me.1.mid	Foo Fighters - Big Me.1	2.0	Alternative	Rock	C	Major	4/4	...	1.0	48.0	Guitar 1, Acoustic Guitar	True	C, E, G, C, G, E, C, C, C, E, G, G, C, C, E, G...	C	4543.0	0.270526	2187.988971	0.981730
1	Foo Fighters	My Poor Brain	Cleaned Lakh\Foo_Fighters\My_Poor_Brain.mid	Foo Fighters - My Poor Brain	2.0	Alternative	Rock	F#	Major	2/4	...	0.0	47.0	Piano, Guitar, Vocal, Bass, Drums	True	F, F, C#, C#, C#, C#, C#, E-, E-, F, F, C#, C#...	C#	4410.0	0.266893	1316.729858	0.823810
2	Foo Fighters	Ill Stick Around	Cleaned Lakh\Foo_Fighters\Ill_Stick_Around.mid	Foo Fighters - Ill Stick Around	2.0	Alternative	Rock	g	minor	4/4	...	0.0	54.0	guitar 1, Electric Guitar	True	G, D, D, G, E-, B-, E-, B-, E-, B-, B-, F, C, ...	G	6321.0	0.234931	1677.496154	0.887518
3	Foo Fighters	Big Me.2	Cleaned Lakh\Foo_Fighters\Big_Me.2.mid	Foo Fighters - Big Me.2	2.0	Alternative	Rock	C	Major	4/4	...	0.0	39.0	Guitar 1, Guitar 1, Electric Guitar	False	C, E, G, C, C, E, G, C, C, E, G, C, G, C, E, G...	C	4286.0	0.262249	1985.426471	0.980635
4	Foo Fighters	Requiem	Cleaned Lakh\Foo_Fighters\Requiem.mid	Foo Fighters - Requiem	2.0	Alternative	Rock	D	Major	4/4	...	0.0	21.0	Electric Guitar	True	A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A...	A	1668.0	0.564149	361.683673	0.981415

5 rows × 22 columns

From this point forward the code uses the sklearn package to perform machine learning operations on the dataset we created. We performed Kmeans clustering to uncover any information on the relationship between "Tonality Percentage" and "Crowdedness". We then integrated that into our classification model by adding a ClusterSimilarity feature that measured its proximity to its portion of the cluster. We used a Pipeline to encode the categorical variables and to scale the numerical variables. After preprocessing the dataframe for the model, the data was then split into train and test portions. We used cross evaluating to analyze the data and get accuracy scores without looking at the test data. We selected a LogisticRegression, RandomForest, and SVC model for comparison. The highest performing score was the random forest score so we selected that one to do a Randomized Search to find the best features for the random forest classifier which came out to be max_features =2 and n_estimators = 75. We then displayed its results in a confusion matrix.

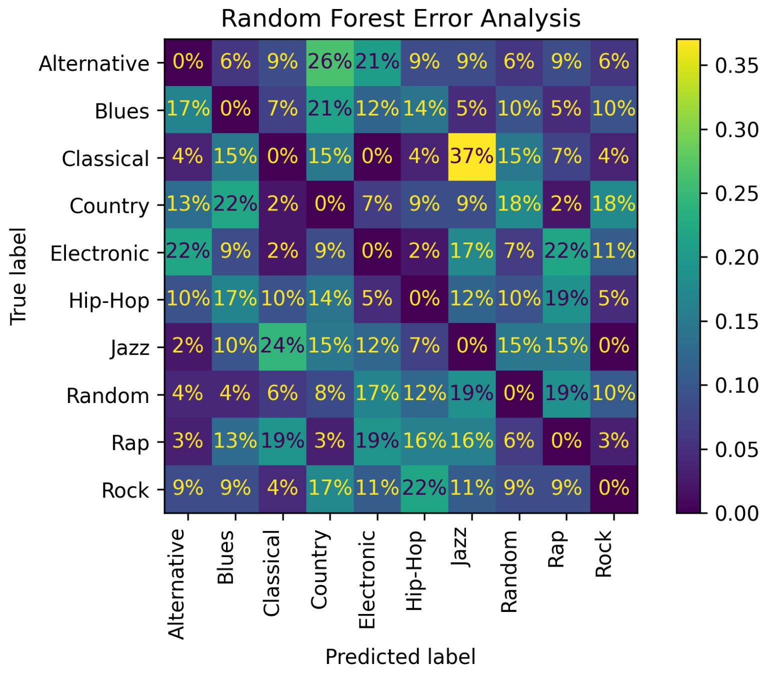
Results

What we found was that the model was not able to correctly classify the genre accurately and reliably. We calculated the precision, recall, and F1 score from the classifier and the came out to be: 0.196, 0.196, and 0.189 respectively. Considering there are 10 genres, this is not awful as the model can predict it more accurately than a random guess, which we would expect to give about 0.1 score for each. However, this is far from a reliable model so we extracted some visualizations to help us see where this unreliability is coming from. We first outputted the confusion matrix of the full data set's predictions.



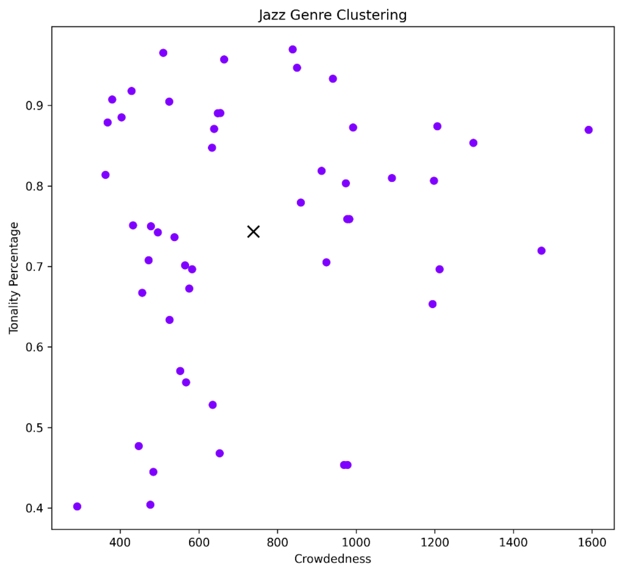
This confusion matrix revealed a few things. Firstly, it revealed that the classifier is not equally good at classifying each genre. It is pretty good at classifying Rap and Classical music, however, it is not very good at classifying the other genres on the list.

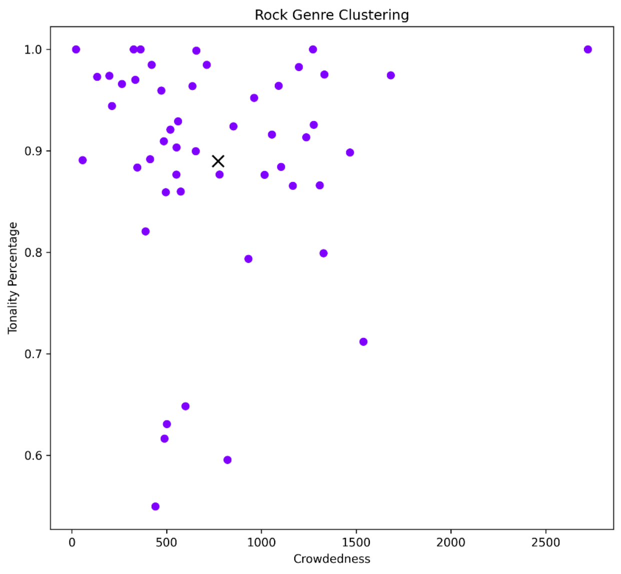
In order to analyze where these errors are happening we created another confusion matrix with sample weights and normalization.



This confusion matrix showed us that when the classifier is making mistakes it is making mistakes understandably. For example, the model never classified a Jazz song as Rock or Hip-Hop because it was generally easy to differentiate between the two. However, it frequently made the mistake of classifying Alternative, Blues, and Rock songs as country. This is understandable because the genres blend with each other pretty frequently.

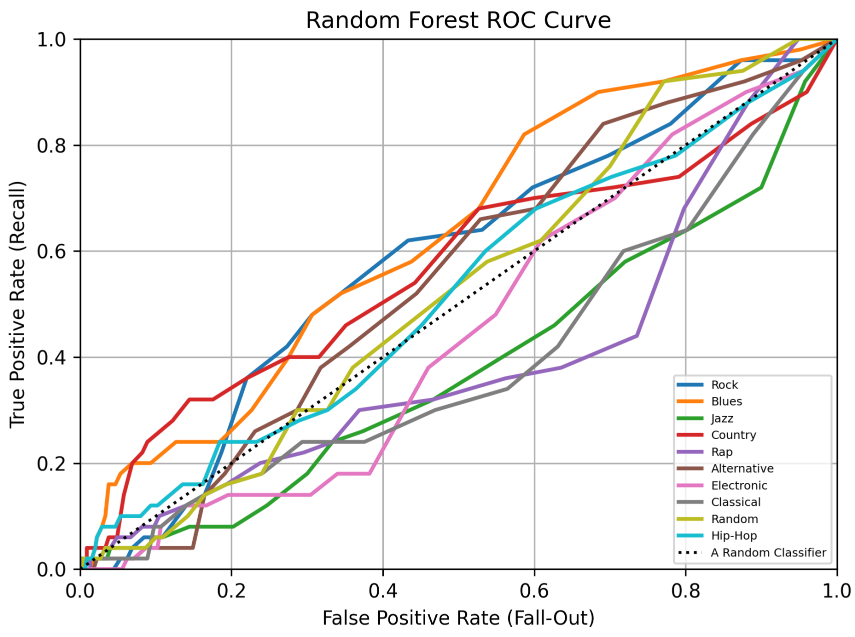
Taking a look at the individual Kmeans clusters of Jazz and Rock we created can show some of the differences between the two genres.





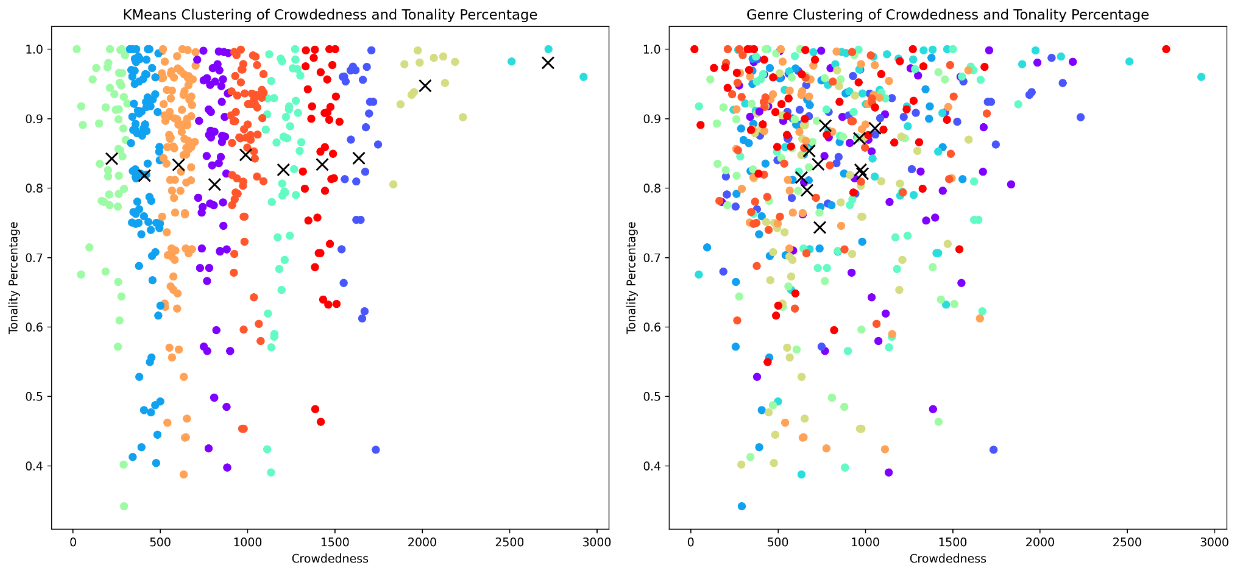
Comparing these two Kmeans cluster reveals a lot about the two genres and shows the relevance of adding the “ClusterSimilarity” feature two the dataset. As expected, the tonality and crowdedness of jazz is significantly lower than that of Rock. The spread of the data also differs between the two. Jazz, characteristically, is a genre with a much looser set of composition rules which leads to the points being further away from the mean on average. Rock typically has a strict way of being written using guitar pentatonic scales and very typical diatonic chords. This leads to the genre having a lot of songs with 100% tonality, whereas with jazz, if it has a high degree of tonality it likely is not a very good jazz song.

In order to perform some further error analysis we created a ROC curve to show the Fallout vs Recall.



Since this curve centers pretty heavily around the random classifier, it is clear that the model has difficulty correctly identifying genres.

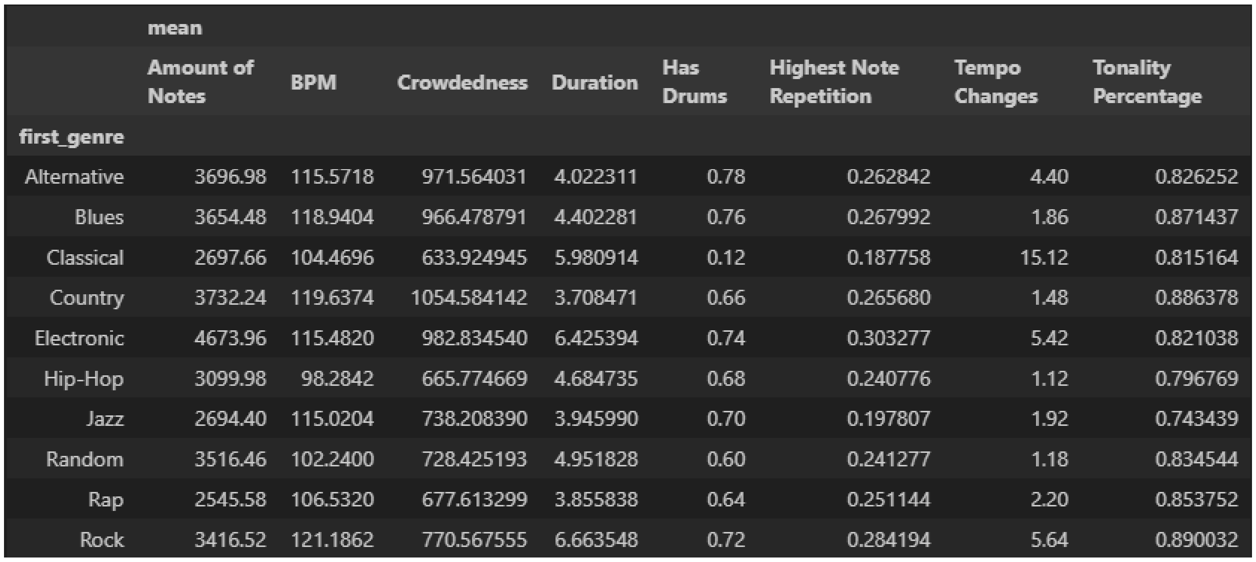
Looking at the full Kmeans clustering performed we can compare how it stacks up against the genre clustering.



Comparing the two plots we can see that the genres do have significant differences in tonality percentage and crowdedness from each other. However, ideally, if the difference was high enough the Kmeans clustering would be able to somewhat cluster the genres accordingly.

This is clearly not the case from the two plots since the genres have a significant amount of overlap.

Finally, we generated a pivot table with the numerical data to see if there were any noticeable differences.



	mean							
	Amount of Notes	BPM	Crowdedness	Duration	Has Drums	Highest Note Repetition	Tempo Changes	Tonality Percentage
first_genre								
Alternative	3696.98	115.5718	971.564031	4.022311	0.78	0.262842	4.40	0.826252
Blues	3654.48	118.9404	966.478791	4.402281	0.76	0.267992	1.86	0.871437
Classical	2697.66	104.4696	633.924945	5.980914	0.12	0.187758	15.12	0.815164
Country	3732.24	119.6374	1054.584142	3.708471	0.66	0.265680	1.48	0.886378
Electronic	4673.96	115.4820	982.834540	6.425394	0.74	0.303277	5.42	0.821038
Hip-Hop	3099.98	98.2842	665.774669	4.684735	0.68	0.240776	1.12	0.796769
Jazz	2694.40	115.0204	738.208390	3.945990	0.70	0.197807	1.92	0.743439
Random	3516.46	102.2400	728.425193	4.951828	0.60	0.241277	1.18	0.834544
Rap	2545.58	106.5320	677.613299	3.855838	0.64	0.251144	2.20	0.853752
Rock	3416.52	121.1862	770.567555	6.663548	0.72	0.284194	5.64	0.890032

As we can see, the classical genre has some large differences from the other genres that likely led it to have a higher recall and precision. The classical genre has a much higher number of tempo changes and a much lower probability of having drums in the song. These features were likely a key part in increasing its accuracy in detecting that genre.

Analysis

The first issue that was encountered was after collecting and cleaning the dataset. The data (i.e. the MIDI files) that needed to be processed were taking a long time to run the functions called from music_processor.py due to the nature of the data files. An idea to address the problem was to have access to a supercomputer dedicated to processing large sets of data with less delay time. Unfortunately, this approach was going to slow down the progression of the project and the decision was made to cut those functions and not include them in the analysis. The effect of not having those functions on our final results was significant since the information retrieved could have been useful in our prediction model.

Discussion

We were able to successfully clean our data set, but only certain genres were able to be correctly classified due to the broad umbrella of many of these genres. We did run into difficulties when attempting to process the MIDI files and experienced extremely long run times with certain functions, which did affect our prediction modeling. We did discover how certain genres, specifically rap and classical, are much more easily classified than others due to distinct differences. However, we hope that this model could be further developed to distinguish between more similar genres, and possibly even generate songs.

A major reason for us as a group to create this project was because we wanted to find some way to think about music programmatically. AI music is not commercially viable due to the high level of creativity it takes to create an authentic sounding piece. The creation of software that can analyze music is a crucial component prior to the development and deployment of finalized neural network models.

References

Data Source:
https://www.kaggle.com/datasets/imsparsh/lakh-midi-clean

Music Genre Dataset (data for matching genre):
https://www.kaggle.com/datasets/thedevastator/spotify-tracks-genre-dataset

Kaggle learn:
https://www.kaggle.com/learn

Music21 Library: ←- This library is necessary for running the code up until the creation of the dataset. If you'd like, the code should run with the existing dataset but only from the point of loading it in from the existing csv file.

https://web.mit.edu/music21/