I divided my project into four parts: Standardized the data, handling NaN, dimensionality reduction and the actual prediction. I will explain these four parts one by one.

Standardizing data:

In this section, I mainly deal with the columns of ‘mode’, ‘key’, and ‘music_genre’. First, I code all of the “minor” in ‘mode’ column to 1 and all of the “major” in ‘mode’ to 0. Second, I code each key from “C” to “B” to 1 to 12. Lastly, I changed all of the music genres into numbers from 0 to 9.

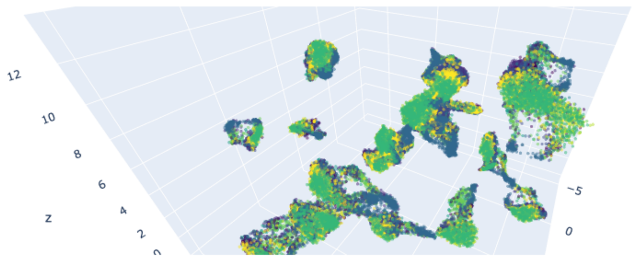
Nan Values:

There are two columns with missing values in this data set: ‘tempo’ and ‘duration_ms’. Initially, I tried to use machine learning algorithms, in particular neural networks to predict the missing ‘duration_ms’ data by the other features in this data set. However, the result was not ideal with a RMSE of more than 8000. In addition, since I still need to reapply the neural network to predict tempo, it would then be too difficult to implement since I need to take care of the overlap problem of missing data. Therefore, for the sake of simplicity, I just use the method of mean imputation to deal with the NaN values in my dataset.

Dimensionality reduction:

First of all, I normalized instead of standardized each column with “outstanding” data such as ‘popularity’ and ‘duration_ms’ which make them fall into a range between 0 and 1. Since some of the columns were not normally distributed, it’s not reasonable to standardize the data. With the categorical data ‘mode’ and ‘keys’, I left the mode as 0 and 1, but the problem came up with ‘keys’. By following the step which I’ve taken in step 2, there were 12 different categories with numbers from 0 to 12. Since every other feature ranged from 0 to 1, I was afraid that the category key would be too deterministic for the cluster result and the final classification. Therefore I used the One-Hot Encoding strategy to manually add 12 more columns into the data set and each one represents a different key. The song with the particular key will have value 1 on its corresponding key column and every other key column remains 0.

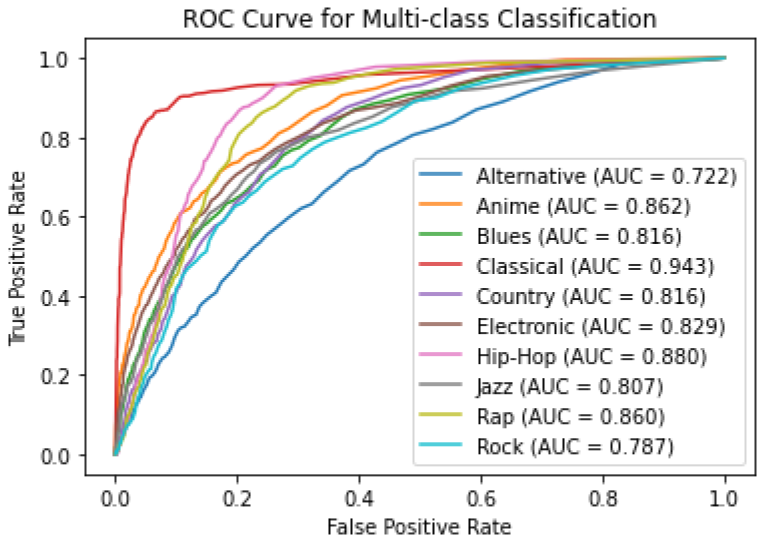
Now, it comes to dimensionality reduction. I tried 2 different methods of dimensionality reduction: T-SNE and Umap. Sadly, none of them gave me a clear separation between each genre. However, compared to T-SNE, Umap does have the advantage of its computation speed. Also, by comparing the AUC score in the last section, I found out that the AUC score for T-SNE with reduced to 3 components was very similar to the Umap with reduced to 4 components. Therefore, in order to make sure that I would get a solid AUC score in the last section by using Umap, I need to reduce it to at least 4 components. In the end, I decided to reduce it to 5 components and I will explain it in the next section. And, this is the graph for the first three components with the reduced data.



Classification:

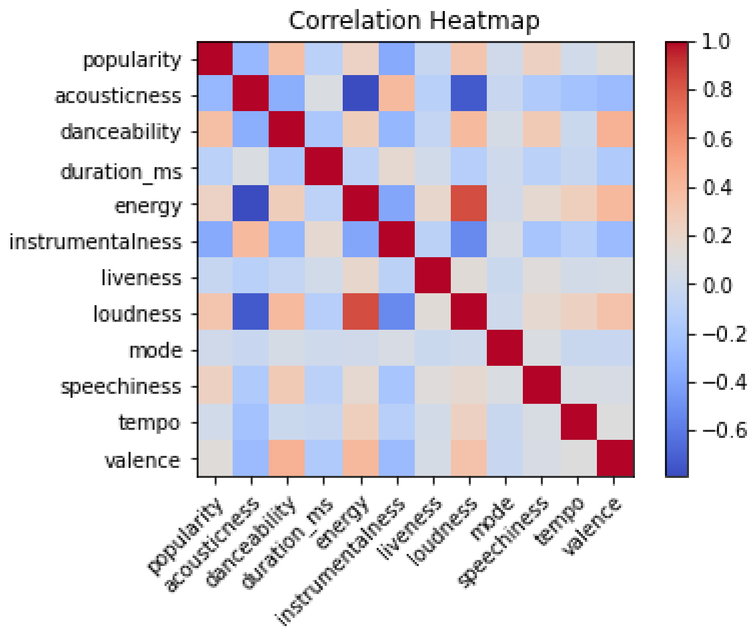
Clearly, from the plot above, we could see that it could not be linearly separable. As a result, I chose to use Random Forest, and there were several reasons. Firstly and most importantly, random forest was fast. It generally took me less than 20 seconds to finish the training. Considering that the data set was quite big with 45000 training data and more than 20 features, other methods such as neural networks might take a long time to train. Secondly, random forests could handle the noise data better. We all know that the boundary between each genre might not be that clear. For example, I anticipated that the boundary between Jazz and Blues would not be clear and it required the model to handle these noise data. By randomly selecting data and features each time, Random Forest could effectively solve the problem. Lastly, I later found out that the random forest could also handle the categorical data well. I first used the One-Hot Encoding data set to run the Random forests and then used the data set with 'key' ranging from 0 to 11. The difference between the results was minute.

Back to the final reduced dimensionality of the data, I chose 5 because it had around 4 percent improvement of the AUC score compared to the dimensionality 3, and 2 percent on dimensionality 4. If I extend to 6 dimensions, there would be no improvement of AUC score.



The Final average AUC is 0.83216.

Interest Findings:



I constructed a correlation heat map for the features and found out that energy and loudness had a really negative correlation with acousticness which make sense. One thing out of my expectation is that popularity had positive correlation with danceability and loudness. It looks like people really enjoy joyful music nowadays.

Another thing was that from my domain knowledge, I don't think 'key' will have any impact on the genre of the songs. However, after I dropped the key column and ran the whole model again, I found out that it impacted my AUC significantly.