General reading notice:

- All the graph is at the every end of the report for the sake of flow and not interrupting the readings.(Graphs are separated into sections)
- All the numbers are generated by python version 3.12 with random seed = 14562458.
- Things that are not in the code but in the report are deleted due to simplicity and clarity of the code.
- Conclusion is after graphing.

Data Cleaning:

- For this capstone project, everything started from the cleaning part of the job. First, I seek for the original dataset which I was looking for the shape and the overall identity of the data. For this dataset, I could see that there are 5 rows that are missing data and in total of 18 columns. As mentioned in the guidance, I dropped the 3 columns that are not as important to us. The code representation is as follows: columns_to_drop = ['instance_id', 'track_name', 'obtained_date']. After dropping the columns that are needed, I focused on cleaning and modifying the missing values. The first thing that caught me is the data in the tempo category. In this column(feature) I first transformed them into numerical data, instead of strings. Then I visited every data that can not be converted and replaced them into NaNs. In this step it is crucial to know in the original dataset, these NaNs were marked as "?". However since we do want to keep this data instead of dispose of it, I performed a median calculation based on music genres. As we found all the median of tempo for each type of music, we insert these medians to the NaNs. This helps me to keep loyal to the original dataset and guarantees that there are not any extreme values that would transform my data points into outliers. Furthermore, I did a drop_NA again just in case to see if I am missing any values. As for rows duplicates, at first I do spot 2 rows that python spots as duplicates. However as I print the two rows out individually, I could see that despite being the same song, they are different versions of the song. Therefore we do keep both of them. Furthermore, I split my data into both numerical and categorical data. And for each type I print out the sum of NAs in each type. Which further confirms my previous concerns regarding NAs. Then, I want to further check and see if the distribution of each music genre is evenly split. I printed out the total genre counts and the number of unique genres. This gives me for all 10 genres, each genre contains 5000 music. After I am sure that my data is cleaned and ready to perform models, I started to build visualization for each category to see if the dataset is normally distributed.

Visualization:

- For visualization, I used 2 methods in total to see how the graph and the data would perform. First as I mentioned above, I used the data I have split into numerical and categorical to plot. In the plotting part of the program I specifically used seaborn instead of matplotlib, this is because I want my graph to be colorful and in this way we could tell specifically how each feature is performed in terms of their music genres. It is interesting to see that for many features, data are mixed together, and no specific music type is left out. However, for some features such as loudness and energy, classical music seems to be isolated. In the numerical datasets, I performed the histoplot to see whether the data is normally distributed, and based on the graph, Acounstiness is strongly skewed to the right. This finding alarms me to focus Acounstiness transformation on the next part of my project. Besides Acounstiness, all other features seem to be normally distributed and very well spread. Then for categorical data, I specifically focused on countplot, which it gives me a general idea of how the data is spread in each category for example, in both key genre and mode, I could see that C# and major are the more prevalent choice of artist compared to other keys and modes.

Acousticness Transformation:

- In Acousticness Transformation, I specifically focused on two ways. First is the log transformation. And the second is square-root transformation. For log transformation, it is most common and widely used for right skewed data, however in this case. Log transformation does not really affect the distribution as much. Therefore, I choose another approach by using the square-root. In this transformation, the transformation was clear and obvious. The shape is more spread out and less extreme near 0. However it still shows a bimodal distribution where both peaks are near 0 and 1. But it is already much cleaner and nicer. It is more balanced and smooth. As a conclusion, the square root transformation did improve the distribution compared to the log transform and the original data — it's less skewed, less sharply peaked, and more usable for models that prefer symmetry. However, it's still not truly normal (not bell-shaped), because the underlying distribution is likely bimodal. Since many models are not influenced by asymmetric data, thus I would stop here. However it is also important to know and notice that in general it is not possible to shape a bimodal data into a normal normal distribution thus I am leaving the data as it is in this shape.

Preprocessing:

- To prepare for the dataset for classification, I first created dummy variables for the categorical features key and mode, converting them into binary indicator columns that represent the presence or absence of each category. This transformation allowed the categorical data to be used effectively in machine learning models. Then I performed a split for test and train, where I split into training and test sets using a 90/10 ratio, with stratification on the target variable music genre to ensure each genre was proportionally represented. All numerical features were standardized using Z-score normalization, transforming them to have a mean of zero and a standard deviation of one—this step was essential to ensure that features were on the same scale before applying dimensionality reduction. I then applied Principal Component Analysis (PCA) to reduce dimensionality and capture the most important variance in the data. Based on the Kaiser criterion, components with eigenvalues greater than 1 were retained, and a cumulative explained variance plot was generated to determine how many components were needed to preserve 95% of the dataset's variance. This preprocessing pipeline ensured that the input data was well-conditioned and suitable for subsequent modeling. In the meantime it would also reduce noise and overfitting. Further I did compare the result for all models, which in general the accuracy drops less than 1% and AUC drops by little, therefore I choose the 95% variance for generalization. There are 4 PCA with eigenvalue greater than 1, however if we only choose 4 PCA then it is not enough data to ensure or support the classification model. As for converting music genre into labeling, this process would not be needed until we have reached the final model: FNN model, it will be specifically mentioned there. For other models, music genre converting to number is not needed, this process would not perform accuracy of classification or AUC of the classification because it is not boosting any of the performance, rather it is a way of making validation easier and simpler for python. Which is not needed for logistic, random forest, and SVM.

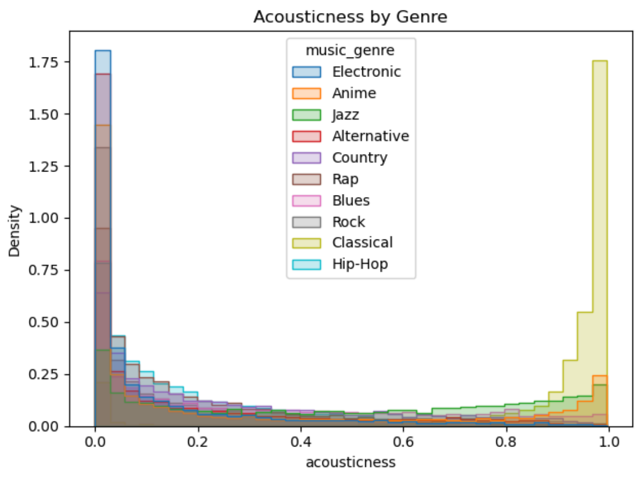
Modeling:

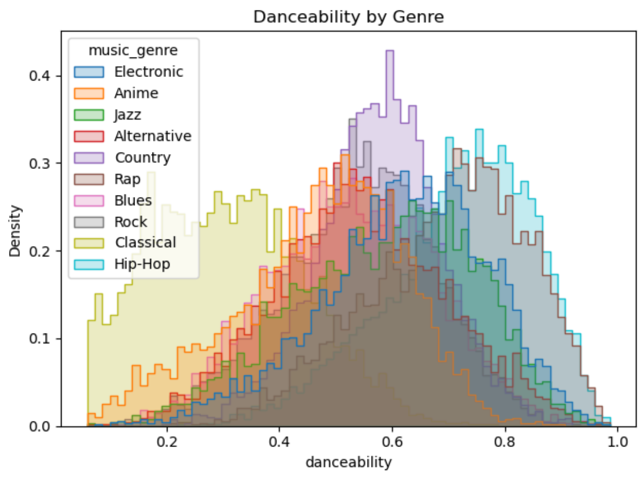
- Model Selection:
    - For model selection, I mainly focused on 4 types of model: Logistic, RandomForest, SVM, FNN.
    - To evaluate the effectiveness of classification approaches for predicting genres of songs. I chose a variety of models including: logistic, random forest, SVM, and FNN using pytorch. This diverse selection was chosen to balance interpretability, nonlinearity handling and performance.
- Logistic:
    - First to start with logistic regression, it served as a baseline model for its simplicity and interpretability. It serves as a useful benchmark because it assumes a linear relationship between the independent variable and the log-odds of the dependent case. Despite its simplicity, Logistic Regression can still perform well on data and provides clear insights. From my model, the logistic regression accuracy is 0.519 with an AUC of 0.898, it is a baseline of mine which also suggested that my model knows the correct answer just not as confident to make it the top 1 prediction. This is due to the fact that we are not predicting a binary result rather a 10 possible result answer.
- Random Forest
    - This model was selected based on the powerful ensemble method capable of capturing non-linear relationships and interactions between features without extensive preprocessing. It is also robust to overfitting due to its use of bagging and decision trees with random feature selection. Since my dataset contains a mixture of transformed numerical variables and dummy variables, Random Forest's ability to handle different types of data effectively and automatically manage feature selection makes it a strong candidate. With RandomForest, I was able to achieve 0.51 accuracy and AUC of 0.897. Which is similar to the previous logistic.
- SVM:
    - For the support vector machine with kernel = “RBF”, I included this model because SVMs are particularly effective in high-dimensional spaces, especially after PCA transformation. The RBF kernel enables the model to capture complex non-linear boundaries between genres. Furthermore, SVMs perform well when there’s a clear margin of separation between classes and are generally less prone to overfitting in low-noise datasets. With results of SVM of accuracy of 0.569 and AUC = 0.920, this indicates that it effectively ranked the correct genre labels higher than incorrect ones. However the top 1 classification has an accuracy of 0.56, this is a significant boost compared to the previous models.
- FNN:

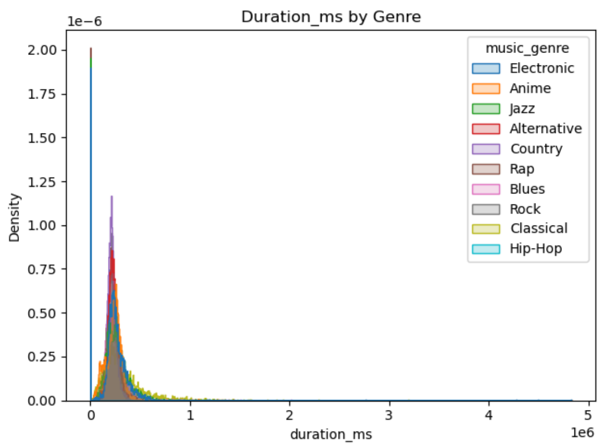
- Final the Feedforward Neural Network (FNN) model, this was chosen to leverage the representational power of deep learning. Unlike traditional models, FNNs can learn intricate, non-linear relationships in the data by stacking multiple layers of neurons. Given the moderate size of the dataset and the reduced feature space from PCA, the neural network could generalize well while still remaining computationally feasible. Additionally, experimenting with FNNs provides insight into whether more complex architectures could significantly outperform classical models in this task. For my FNN, it was tuned with two hidden layers of 64 neurons and ReLU activation, seeking a balance between learning capacity and generalization. Then the use of Adam optimizer allowed for efficient convergence with minimal manual tuning. By applying a LogSoftmax output and negative log-likelihood loss, the model was well-suited for multi-class classification, yielding a good AUC and accuracy score. The output of the FNN was the highest and best performed among all other models I have used, with accuracy of 0.572 and AUC of 0.923, this showcases the importance of deep learning. In this model, I have turned all music genres as numbers for the need of torch do not take in strings, thus the converting part is essential for making good predictions and evaluating the model performance.

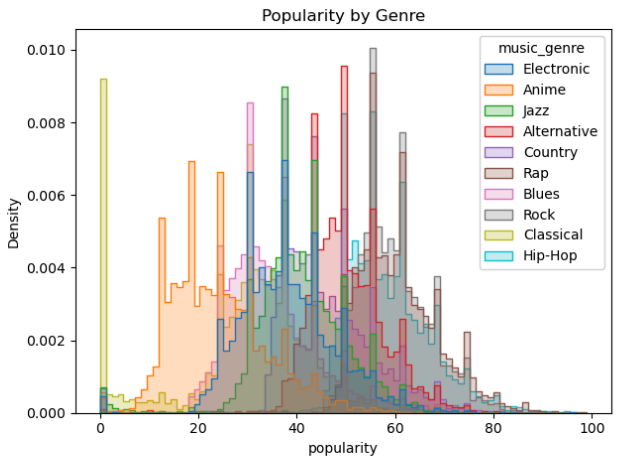
Graphing:

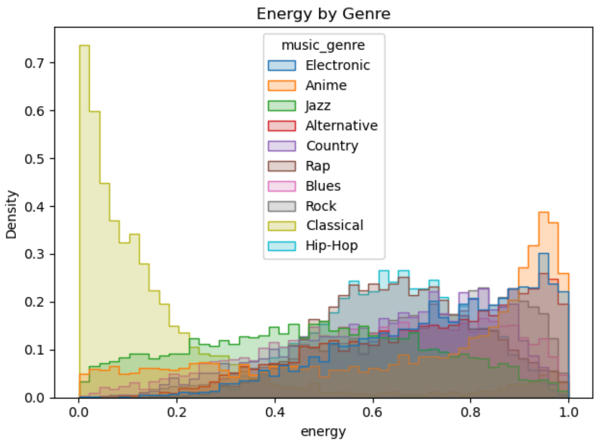
- Visualization:

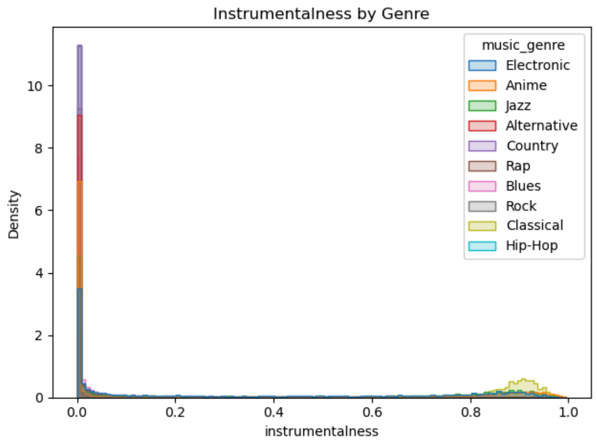


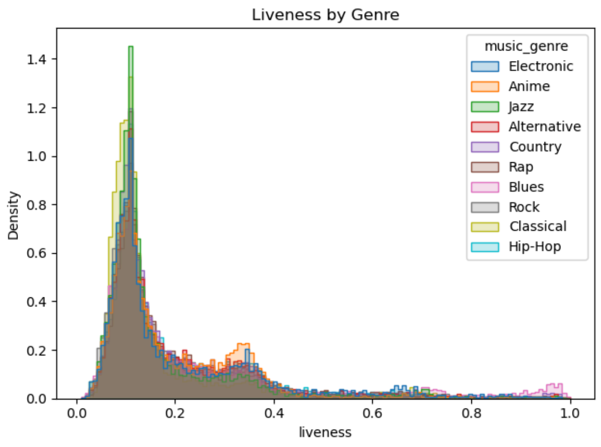


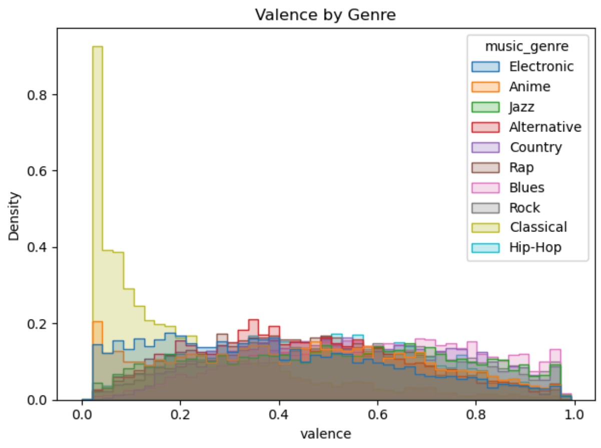


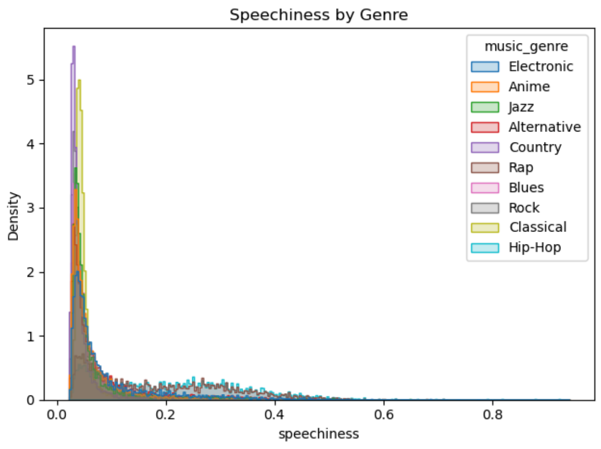


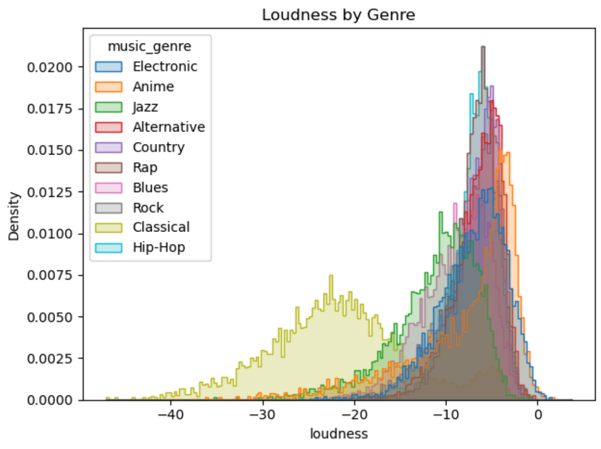

Energy by Genre
music_genre
Electronic
Anime
Jazz
Alternative
Country
Rap
Blues
Rock
Classical
Hip-Hop
Density
energy

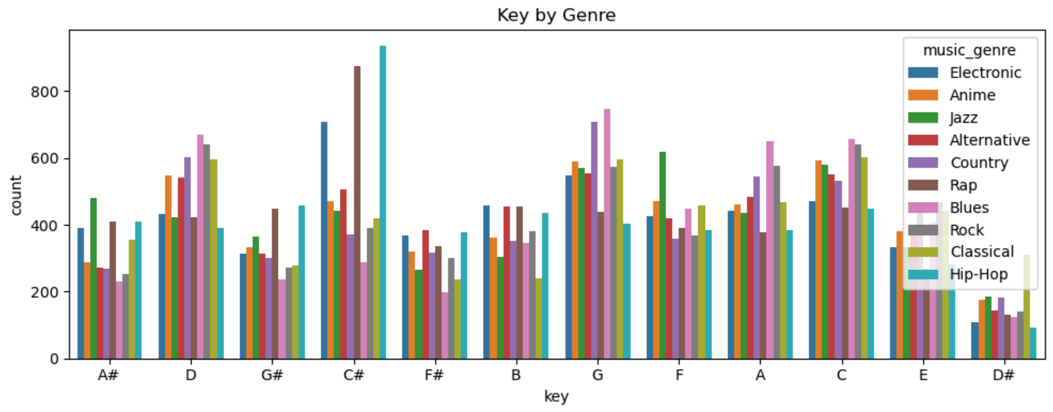

Instrumentalness by Genre
music_genre
Electronic
Anime
Jazz
Alternative
Country
Rap
Blues
Rock
Classical
Hip-Hop
Density
instrumentalness

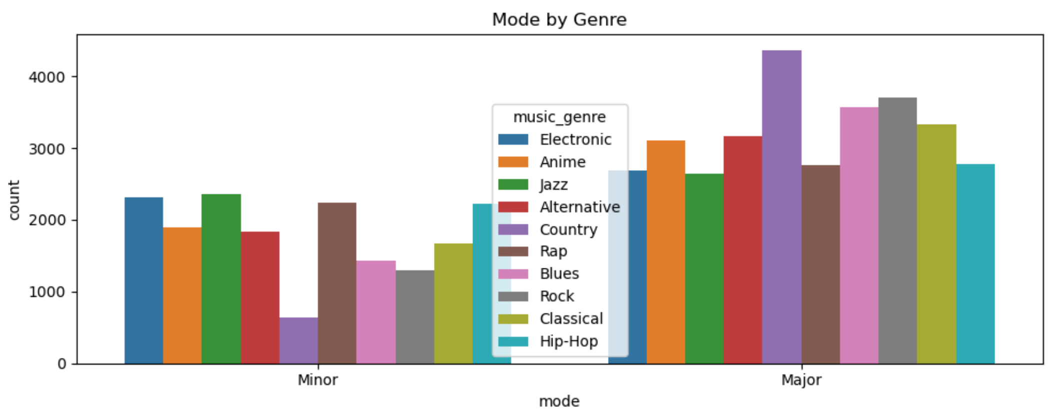

Liveness by Genre
music_genre
Electronic
Anime
Jazz
Alternative
Country
Rap
Blues
Rock
Classical
Hip-Hop
Density
liveness

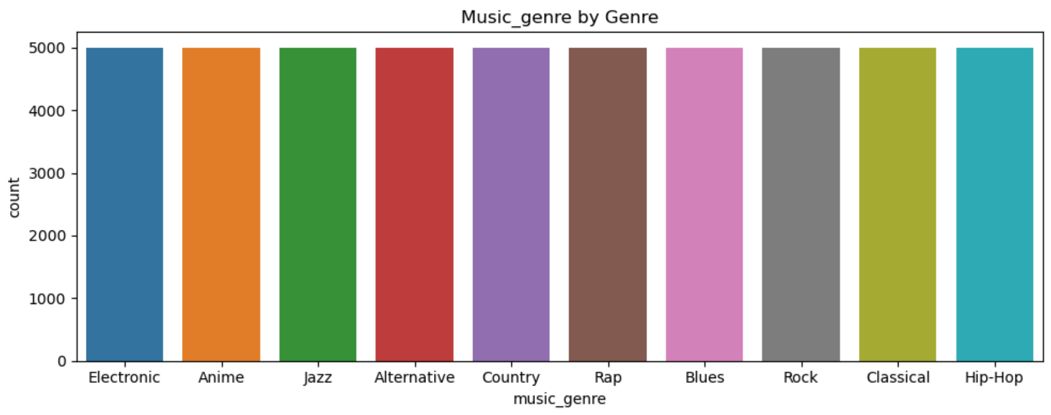

Valence by Genre
music_genre
Electronic
Anime
Jazz
Alternative
Country
Rap
Blues
Rock
Classical
Hip-Hop
Density
valence


Speechiness by Genre
music_genre
Electronic
Anime
Jazz
Alternative
Country
Rap
Blues
Rock
Classical
Hip-Hop
Density
speechiness

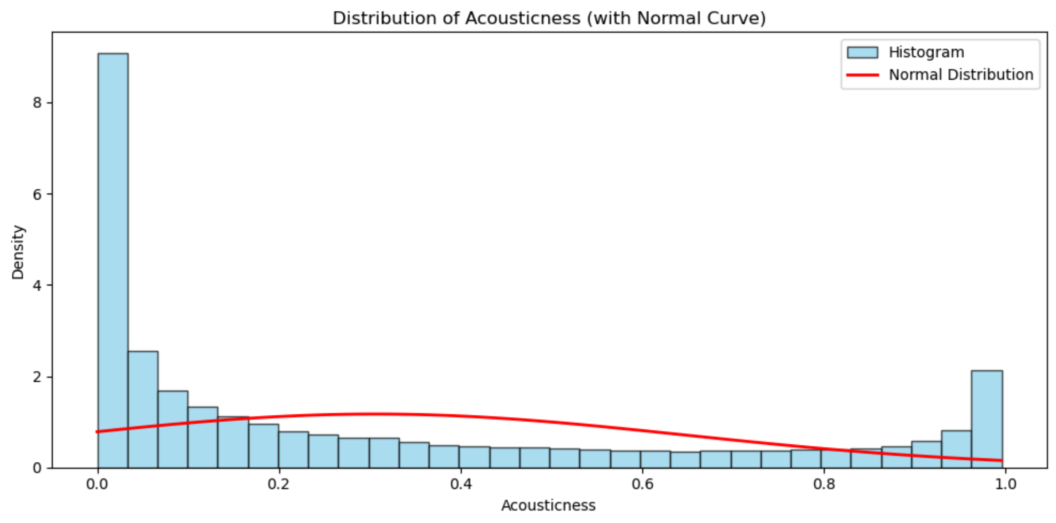

Loudness by Genre
music_genre
Electronic
Anime
Jazz
Alternative
Country
Rap
Blues
Rock
Classical
Hip-Hop
Density
loudness

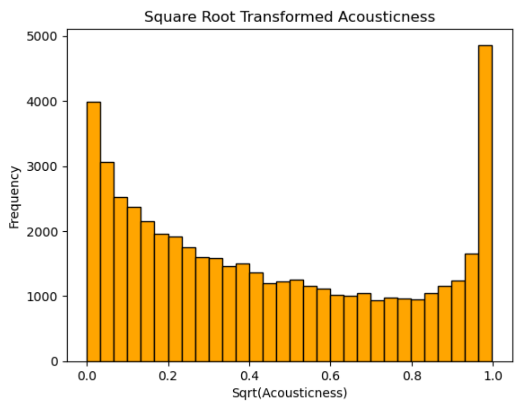

Key by Genre
music_genre
Electronic
Anime
Jazz
Alternative
Country
Rap
Blues
Rock
Classical
Hip-Hop
count
key
A#
D
G#
C#
F#
B
G
F
A
C
E
D#

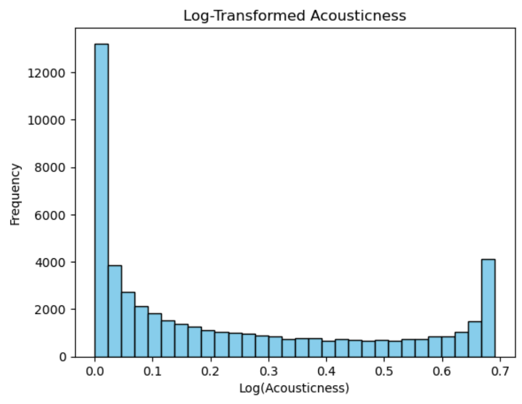




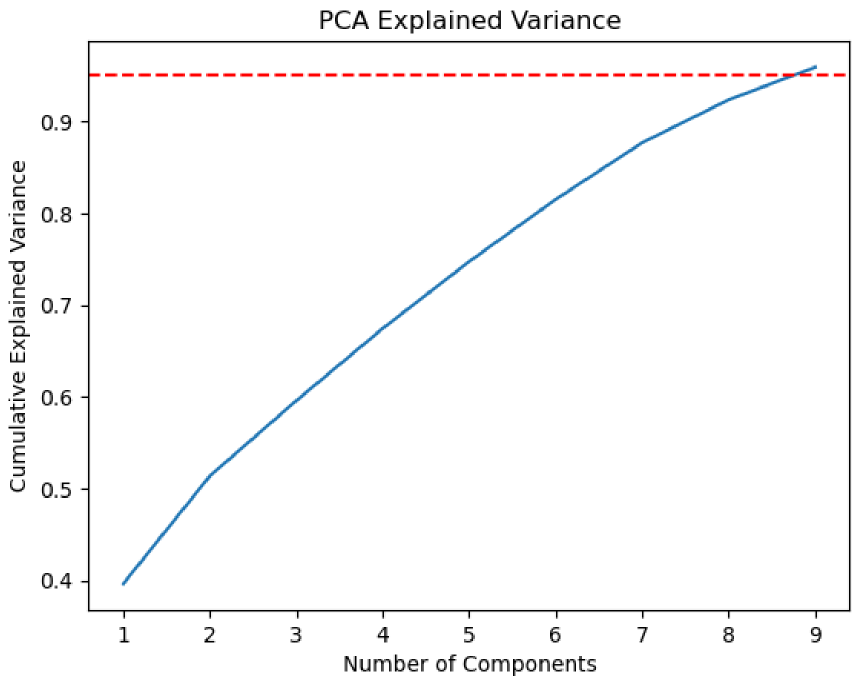
- Acousticness Transformation:







- **PCA:**


PCA Explained Variance
Cumulative Explained Variance
Number of Components
0.4
0.5
0.6
0.7
0.8
0.9
1
2
3
4
5
6
7
8
9