DataProcessing:

I load the music dataset from the 'musicData.csv' file into a DataFrame. Then I replace any '?' characters in the dataset with NaN values and then remove any duplicate rows and rows with missing values. I replace the values of the "mode" column with binary values 0 or 1 to indicate whether the song is in a major or minor key. I also encode the "key" column using the LabelEncoder function from scikit-learn to transform the categorical values into numerical values. Then I convert the genre labels into numeric values from 0 to 9. Also, the column tempo is converted to float type. This ensures that the model can learn from the data and make accurate predictions for new songs.

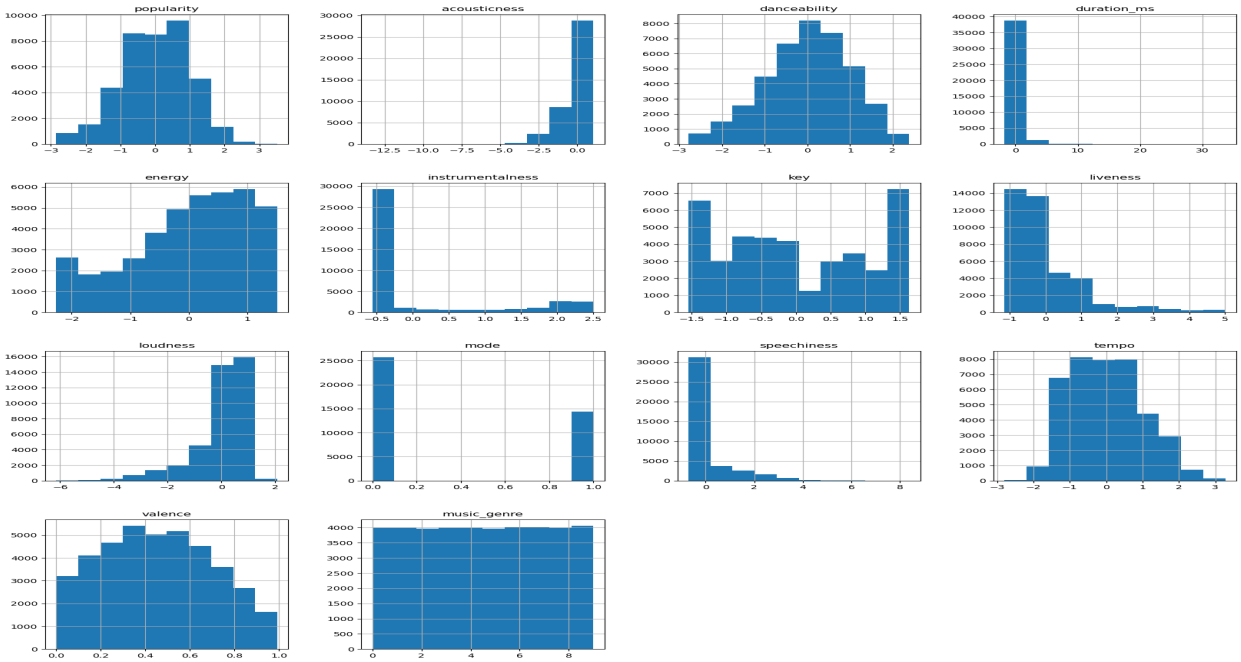
Then I separate the feature variables (X) from the target variable (y), which is the music genre. The feature variables exclude the instance ID, artist name, track name, obtained date, and music genre, Label columns. I split the dataset into training and testing sets, with train_test_split being called with a test size of 500 for each genre, using a random state of 13072967 for reproducibility. I use stratify parameter in train_test_split() to ensure that the target variable is approximately equally represented in both the training and testing sets. This helps to avoid any potential bias or overfitting that could arise if one of the classes in the target variable is overrepresented in one of the sets. The complete test set will be 5000x1 randomly picked genres (500 from each genre). I then normalized the numerical features for both training and testing sets.

I normalize all the numerical features to ensure that the data is in a suitable format for machine learning algorithms to work with and that the features are on a similar scale to prevent bias in the model. Since the acoustic features are unlikely to be normally distributed, I convert the acoustic feature from a non-normal distribution to a normal distribution by log normalizing the data. I avoid normalizing categorical values (like mode) for the purposes of doing dimensionality reduction.

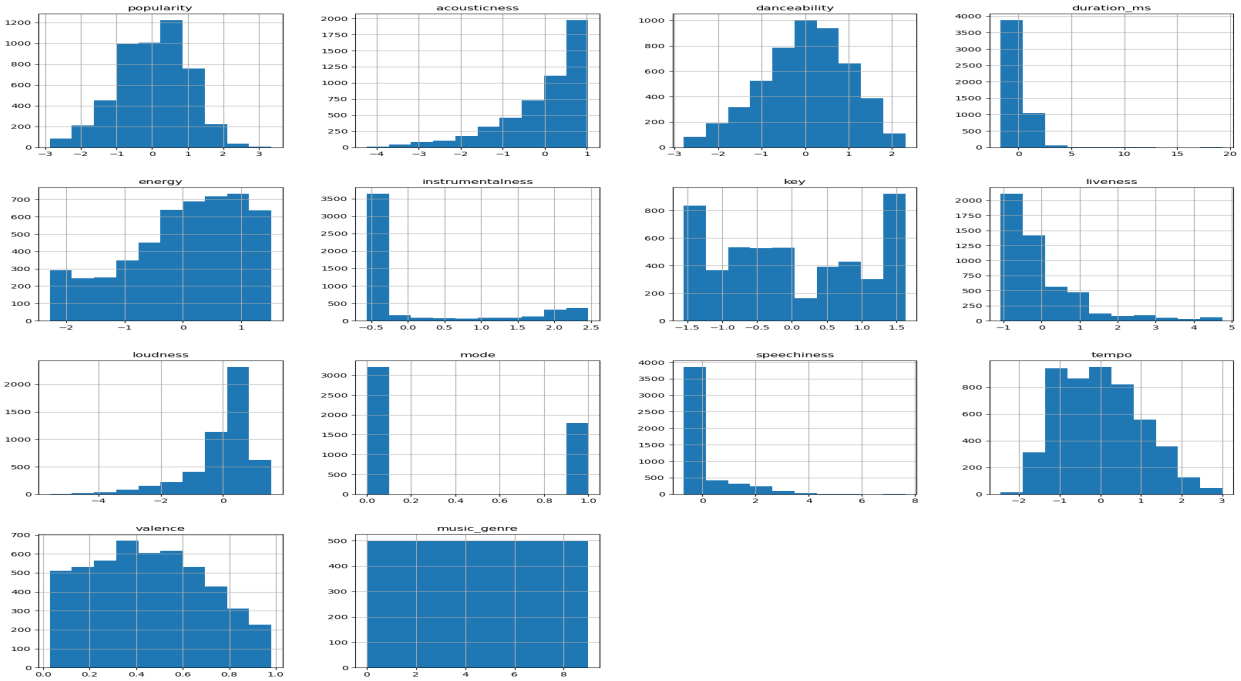
Note that I perform feature scaling after splitting the data into training and testing sets. This is because if we scale the entire dataset before splitting, we could potentially introduce some information leakage from the test set into the training set, leading to overfitting and inaccurate model performance. By performing scaling on the training set only, we ensure that we are not using any information from the test set to inform our scaling decision. Then, we can use the scaling on the test set. This approach ensures that our model is evaluated on data that is representative of real-world scenarios, which can help us make more accurate predictions.

Below is the distribution of all features for training and testing sets:

Training Set:

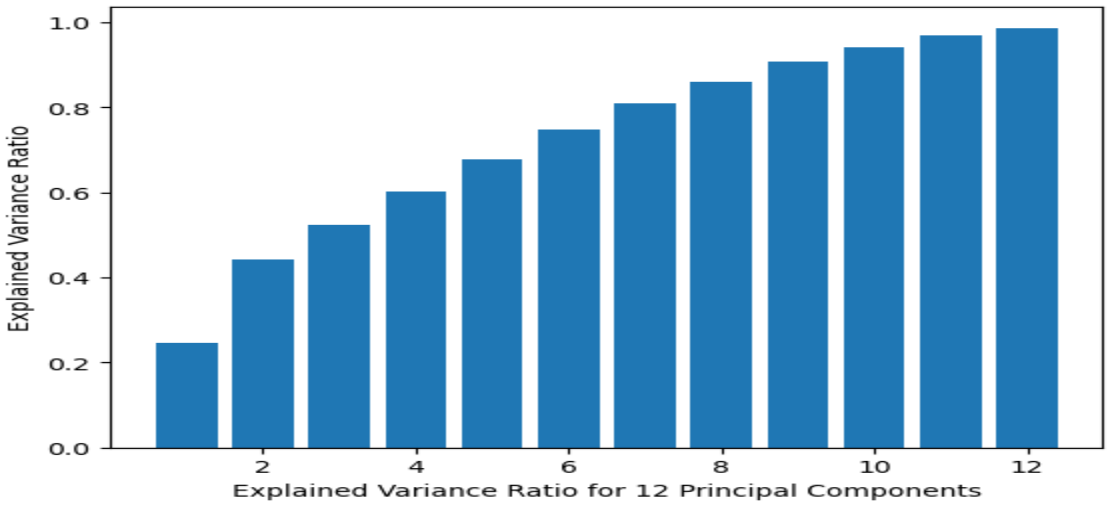


Testing Set:

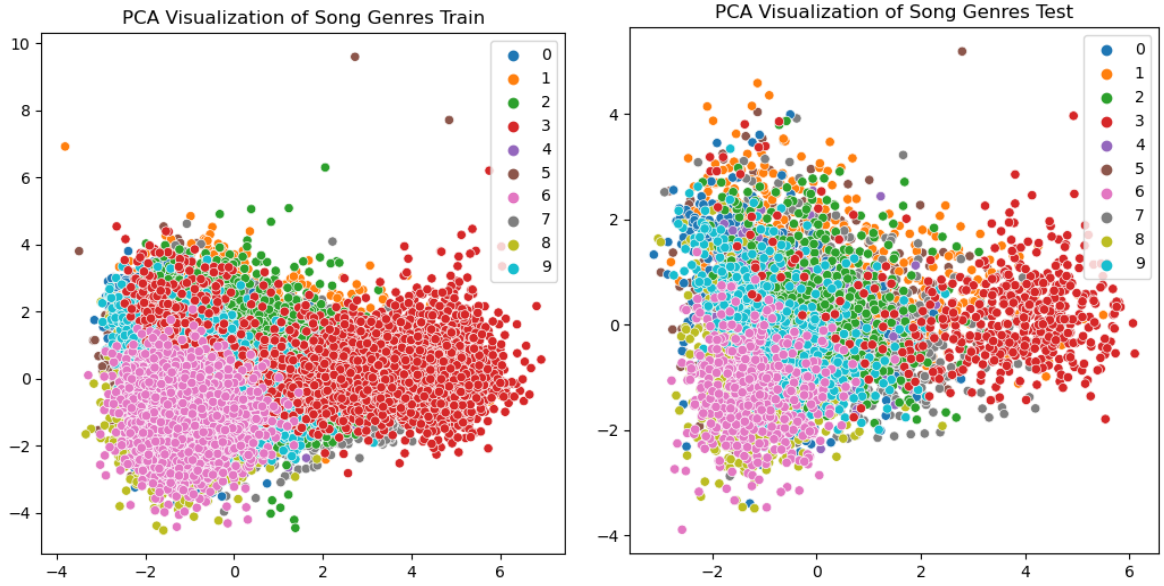


Dimensionality Reduction and Clustering:

PCA



I apply PCA to help improve the performance of the classification model by reducing the noise and redundancy in the data and preserving the global structure of the data. This can lead to faster and more efficient training of the classification model, as well as potentially improving its performance by removing noise or irrelevant features.

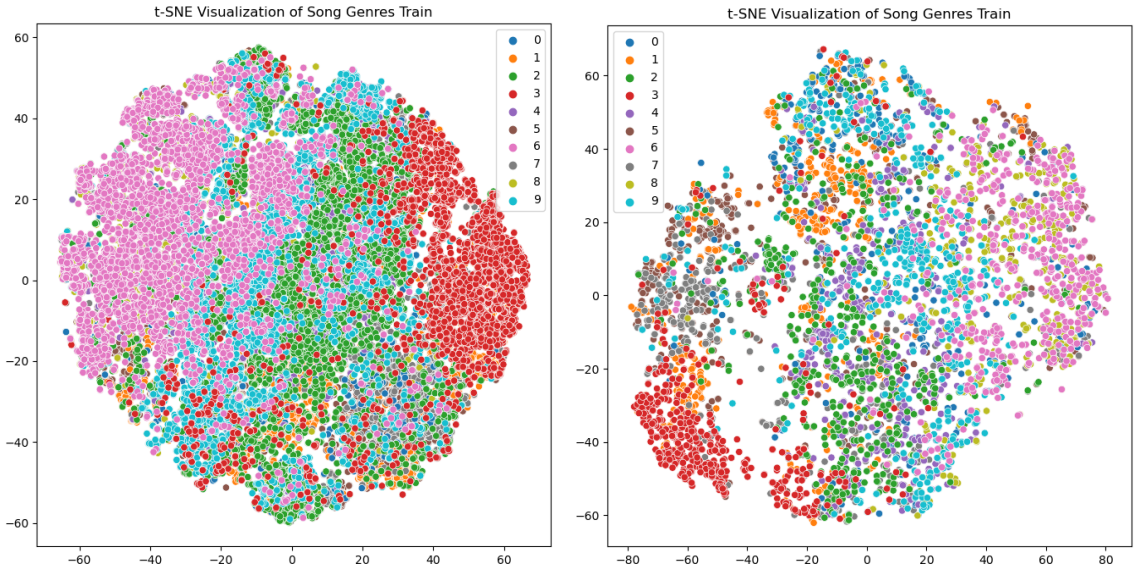


Here we visualize after applying PCA to X_train and X_test with labels from the original dataset and reduced them to 2 dimensions explaining about 40% variance of the original dataset. We

see that dots with the same color are distributed close to each other, and the distributions of X_test and X_train are similar after performing PCA to reduce them to two principal components, indicating that the two datasets share common patterns and relationships between the features. This similarity can be beneficial in building a classification model as it suggests that the model may be able to make accurate predictions on new data and suggests that there is no leakage between the training and testing data.

T-SNE

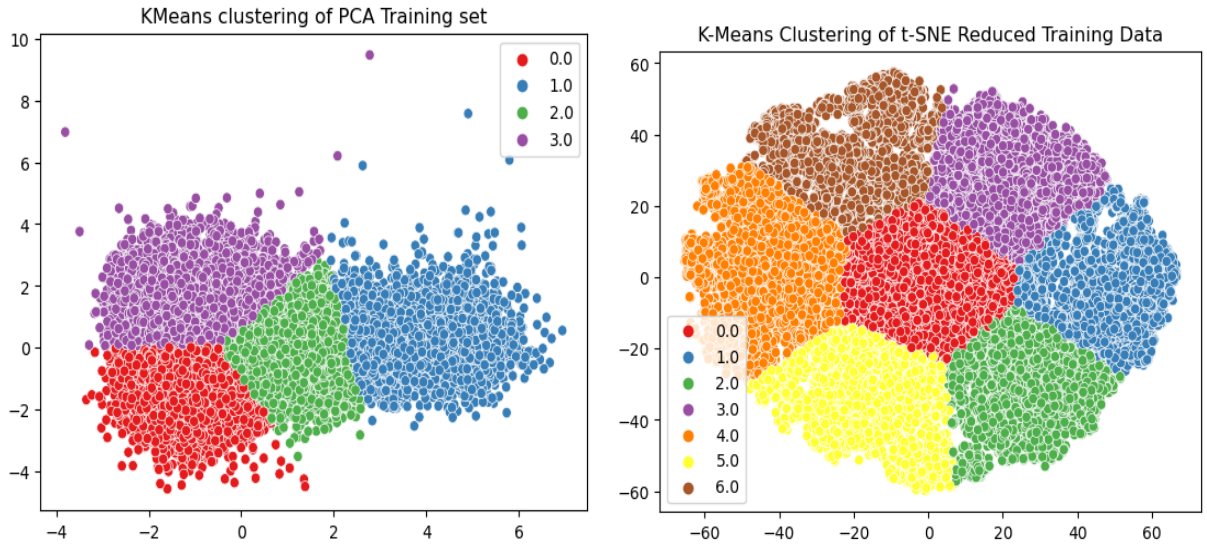
I apply t-SNE to reduce the dimensionality of the data into 2 dimensions for visualization purposes and understand the local structure of the data. This allows us to visualize the data in a lower-dimensional space while preserving the similarities and relationships between the data points.



We visualize after applying t-Sne to X_train and X_test with labels from the original dataset and reduced them to 2 dimensions. We can see that most of the data in the same genres are clustered together in the visualization of the reduced dimensionality. This is an indication that the reduced-dimensional space captures some of the underlying structure of the data, and it is likely that the classification model will perform well on this reduced-dimensional space.

After dimensionality reduction, I then cluster the data reduced by PCA and t-sne using k-means with different numbers of clusters. I calculated the silhouette score for each clustering and plot the results for comparison. The method with the highest silhouette score for the optimal number of clusters can be considered the better method for clustering the data.

I cluster the data reduced by t-sne and pca by k-means with their optimal number of clusters.



The best number of clusters is 4 for PCA with a silhouette score of 0.40.
The best number of clusters is 7 for tSNE with a silhouette score of 0.38.

Based on the silhouette scores and the clustering results, it appears that both PCA and t-SNE are able to cluster the data into 4 distinct groups. This suggests that there may be clear separations between the data points in the feature space. PCA has a slightly higher silhouette score than tSNE, which indicates that the clustering results from PCA may be more reliable than those from t-SNE. Thus, it is reasonable to choose PCA for clustering, as it appears to provide slightly better results in terms of silhouette score. Then, we apply the clustering labels to the X_train and X_test as an additional feature

Classification and Conclusion:

Model selection and Hyperparameter tuning:

During the model selection process, we trained and tuned classifiers. For each model, we defined a grid of hyperparameters and used cross-validation to search for the combination of parameters that yielded the best performance on the training data.

After tuning the models, we evaluated their performance on the test data using both ROC AUC score and accuracy. The best hyperparameters for the RandomForestClassifier were found to be 'max_depth': 10, 'min_samples_split': 10, 'n_estimators': 150. This configuration achieved a ROC AUC score of 0.92 and an accuracy of 0.55. For the MLPClassifier, the optimal hyperparameters were found to be 'activation': 'logistic', and hidden_layer_sizes': (40, 25, 15). This configuration achieved a ROC AUC score of 0.94 and an accuracy of 0.57. The AdaBoostClassifier achieved the best ROC AUC score of 0.87 with a combination of hyperparameters 'learning_rate': 0.5, and 'n_estimators': 100. However, its accuracy was found to be lower than the other two algorithms, with a score of 0.52.

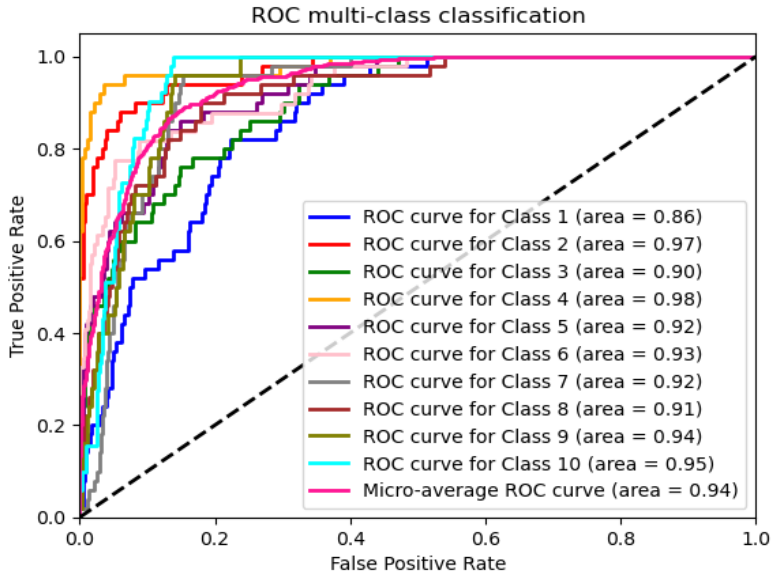
In summary, normalization improved the performance of PCA in dimensionality reduction for genre classification. Applying dimensionality reduction, clustering labeling, and hyperparameter tuning techniques generally improved the classification models' ROC AUC scores and accuracy scores. The idea is that the clustering algorithms (K-means) might capture some useful underlying structure in the data, which can then be used as additional information by the classifier.

Before implementing these techniques, the Neural Network model had the highest ROC AUC score of 0.93, an accuracy of 0.58, the SVM has a ROC AUC score of 0.89, an accuracy of 0.50, the Random Forest and AdaBoost models scored 0.92 and 0.87, and an accuracy of 0.55 and 0.48, respectively.

After applying these techniques, the Neural Network model continued to outperform the other models, achieving a ROC AUC score of 0.94 and an accuracy of 0.59, the SVM has a ROC AUC score of 0.88 and an accuracy of 0.51, the Random Forest has a ROC AUC score of 0.92 and an accuracy of 0.55, and AdaBoost has ROC AUC score of 0.87 and an accuracy of 0.52.

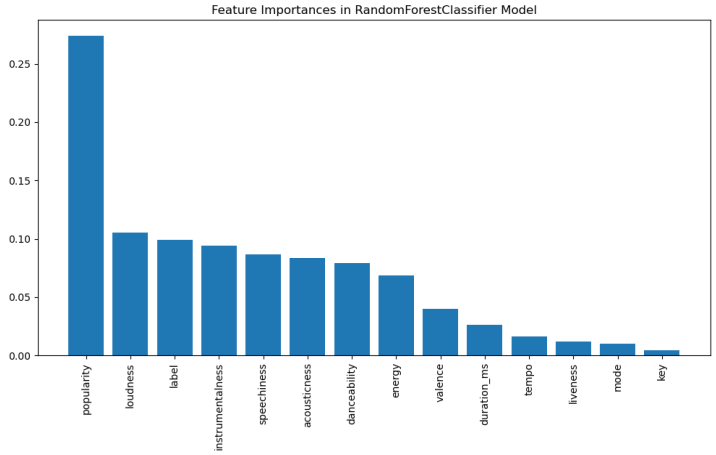
Results:

Therefore, the Neural Network model is selected as the best classification model for genre classification. As a result, we achieved an AUC of 0.94, as shown in the ROC curve graph.



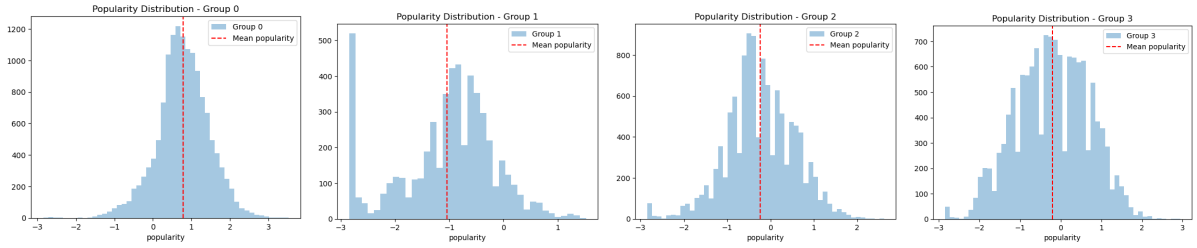
Extra Credit:

We fit a RandomForestClassifier model on the training data with the best hyperparameters obtained from hyperparameter tuning. We then extract the feature importances from the model using the feature_importances_ attribute. Finally, we plot the feature importances in descending order using plt.bar(). This plot shows which features were the most important for accurate genre classification, providing insights into which features are the most relevant for distinguishing between different genres of music.



We can see the popularity of songs is the most important feature. One possible explanation for why the "popularity" feature is important for genre classification is that it may be indicative of the overall style and commercial success of a song. For example, songs that are more popular may be more likely to belong to certain genres that are more mainstream or have a wider appeal. Additionally, certain genres may be more likely to produce popular songs, which could also contribute to the importance of the "popularity" feature in our model.

Then I plot the popularity distribution for each group, we use a histogram to visualize the distribution of popularity values.



The mean and the distribution are quite different from each other for the groups.
We use metrics to further determine the difference in popularity among groups. The ANOVA F-statistic of 8122.97 and a p-value of 0.0 suggest that there is a significant difference in popularity among the clusters. The F-statistic represents the ratio of the variance between the group means to the variance within the groups.

A large F-statistic and a small p-value indicate that the variation in the data is mainly due to the differences between the groups rather than random variation within the groups. In this case, the small p-value of 0.0 suggests that the probability of observing such a large F-statistic under the null hypothesis (that there is no difference in popularity among the clusters) is very low. Therefore, we can reject the null hypothesis and conclude that there is a significant difference in popularity among the clusters.

The evaluation of the difference in popularity among clusters can help in the classification task as it provides insights into the features that are most important for predicting the genre of a song. In this case, we found that there is a significant difference in popularity among the clusters, indicating that popularity may be a critical feature for predicting the genre of a song.