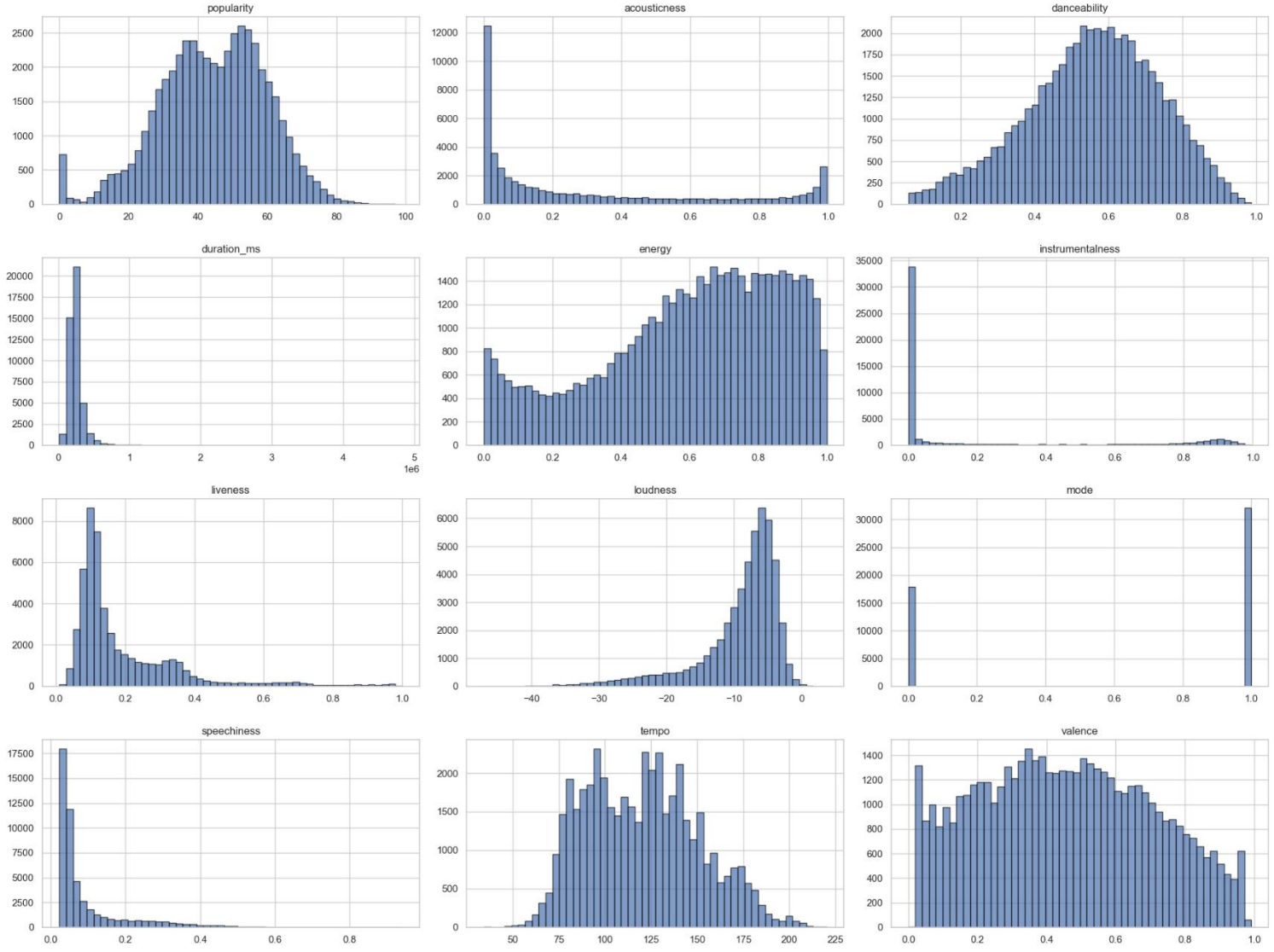
Music Genre Classification Report

Data Cleaning

I first found and removed 5 completely empty rows, which are the only rows with empty values in the dataset. I removed the obtained date, artist name, track name, and instance id columns because they are not useful in music genre prediction. I used a label encoder to encode music genres into numerical labels so I can use them in my classification model. I also mapped the music mode (Major or Minor) to 0 and 1 so that they are in numerical format. I applied one-hot encoding to the songs' keys (A#, A, B, etc.) as they have no ordinal relationship. I also removed one column of the encoded keys to avoid the dummy variable trap and multicollinearity.

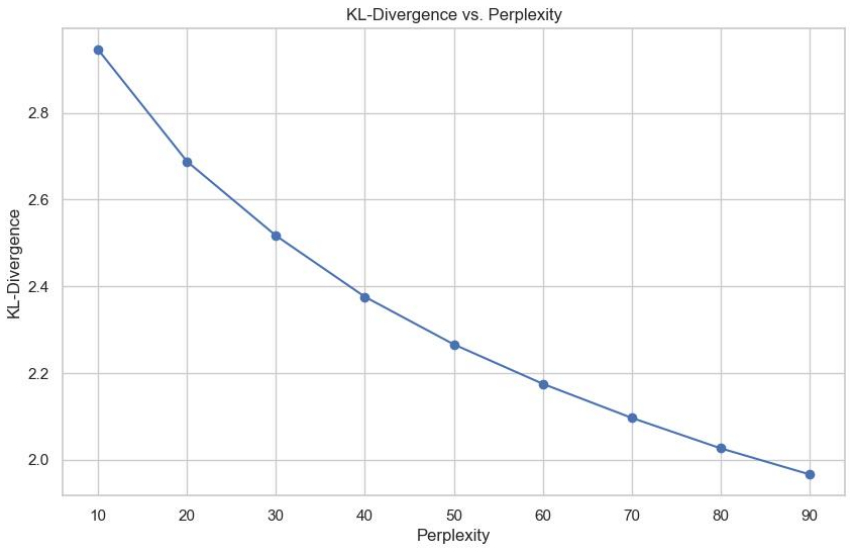
When checking all other remaining features for missing values, I found that the tempo column consists of many "?" values, indicating absence of evidence. I also found that the duration_ms feature consisted of many -1 values, which also represents absence of evidence. I converted all "?" and -1 values to NaN to represent missing data. As there are not many missing values, I imputed them with their column's median to handle missing data in a robust and simple way. Finally, I checked the features' distributions, and excluded the keys in this process as they have meaningless distribution graphs. Below are the distribution histograms:



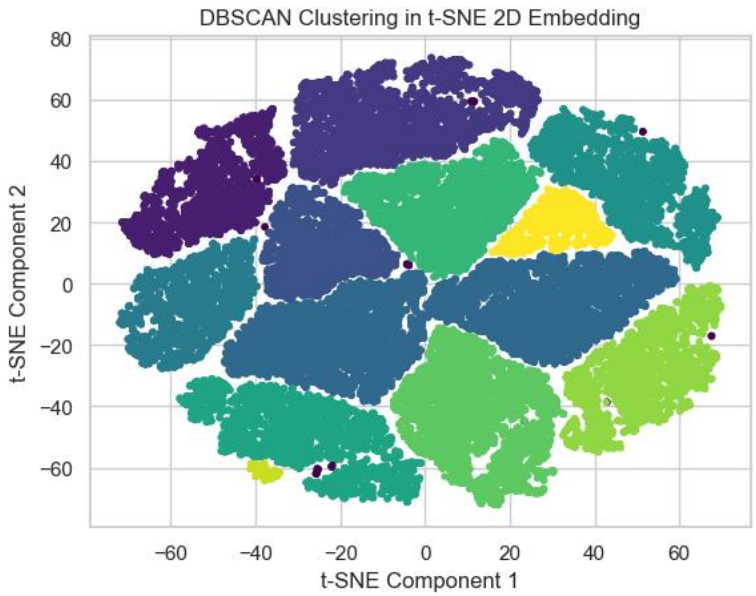
From the histograms, I concluded that none of the features are normally distributed. This is a crucial insight as it means I cannot use methods that assume a normal distribution of data, such as LDA.

Dimension Reduction

Before dimension reduction, I standardized the numerical data so that they are on the same scale. I left out the categorial feature, mode, when standardizing and added it back after to maintain its interpretability and meaning. I tuned the perplexity value for T-SNE using KL-Divergence and found the optimal perplexity of 10 with the highest KL-Divergence value.



I used T-SNE with perplexity of 10 to create a 2D embedding. With the embedding, I initially tried to cluster with K-Means, but the clusters were not appropriate due to their spherical nature. Therefore, I used DBSCAN to cluster instead. I first tuned the epsilon and min_sample values by iteratively trying a range of each hyperparameter value and comparing their silhouette score. Using the optimal epsilon value of 2 and min_samples value of 15, I clustered with DBSCAN and yielded the following clusters:



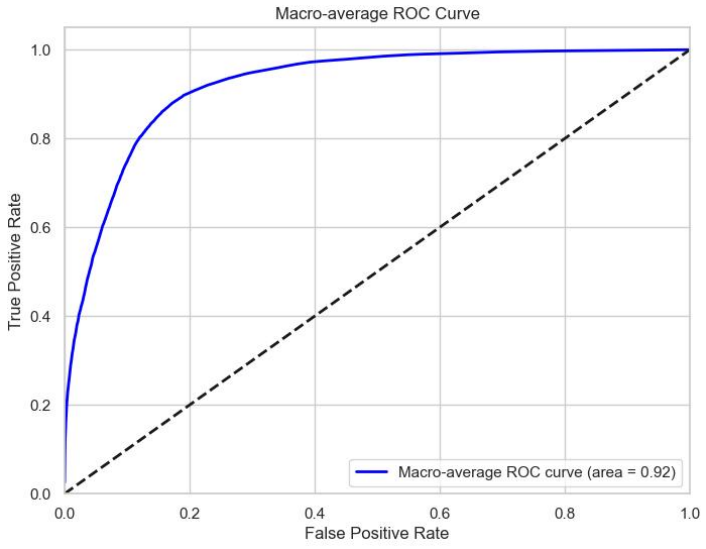
As shown from the graph above, DBSCAN yielded 12 clusters in the t-SNE embedding. As there are 10 genres in the dataset in total, this suggests that there are more types of music than 10. This could be true as the genres could be broken down into subgenres. Moreover, the clustering is not perfect as one can argue the navy-blue cluster could be 2 clusters and that the bottom left cluster should be a part of the cluster above it. Nevertheless, DBSCAN formed reasonable clusters given the positions of points in the t-SNE embedding.

Model Training and Evaluation

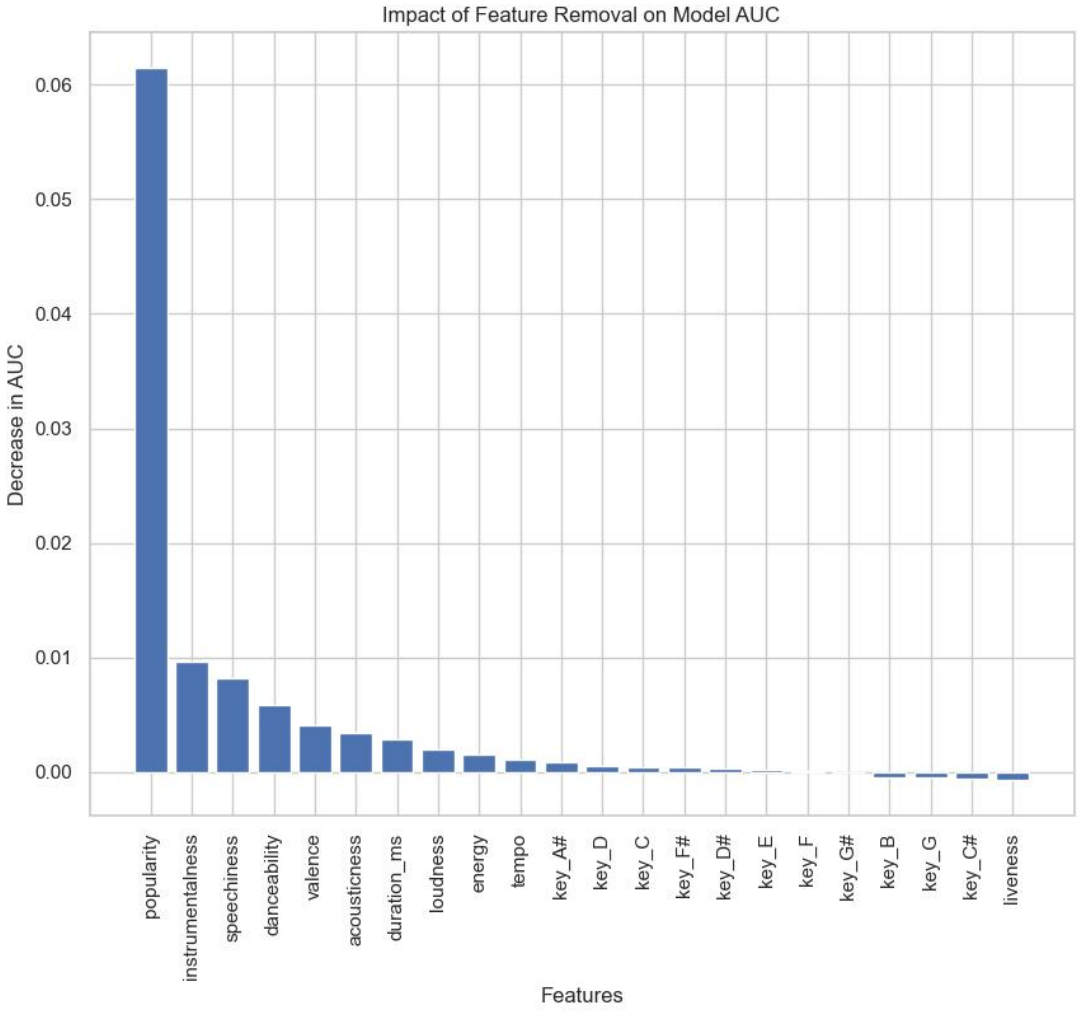
To avoid leakage and overfitting, I split every genre with a 90/10 training split and combined the individual train and test sets. As a result, this yielded a 45000/5000 split that balances the number of values from each genre.

I used a random forest classification model using the raw data to predict genres. I did not use the lower dimension data as using the original predictors resulted in better performance. This might be because random forest can assess the importance of features directly through the training process, allowing it to focus on the most informative features and making it particularly effective when dealing with raw, high-dimensional data. Moreover, the number of raw features is not too high, making it manageable for the random forest classifier.

I started with tuning the random forest classifier's hyperparameters as ensemble methods work best when their hyperparameters are optimized. I held n_estimators at 100 during tuning because adding more trees did not further increase the AUC. I only tuned max features and max samples because my initial tuning indicated that it is best to leave other hyperparameters unchanged. Using GridSearchCV, I tuned the hyperparameters using 1/5 of the training set for cross validation in each iteration. I evaluated every iteration using the AUROC because it is one of the best metrics to assess classification models, especially when there are no class imbalances. The optimal hyperparameters were max_features = 0.4 and max_samples = 0.2, which yielded an AUC of 0.923 in the validation set. I used the optimal hyperparameters to build the random forest model and evaluated it using the test set. I used the macro-average AUC to assess the model's performance because it weighs all classes equally. The final macro-average ROC AUC calculated from the testing set is 0.92.



To measure which features impact the model's performance the most, I iteratively removed each feature from the model and assessed the model's performance without it using macro-average AUC. The following graph shows the predictors' impact on the model:



As shown in the graph, popularity impacted the random forest classifier's performance the most, dropping the AUC by over 0.6. The other predictors only dropped AUC by less than 0.1 when removed, highlighting the importance of popularity as a predictor of music genre.

Final AUC: 0.92