Final Project: Classification flavor for Music Genre

Made by Bowen Jiang

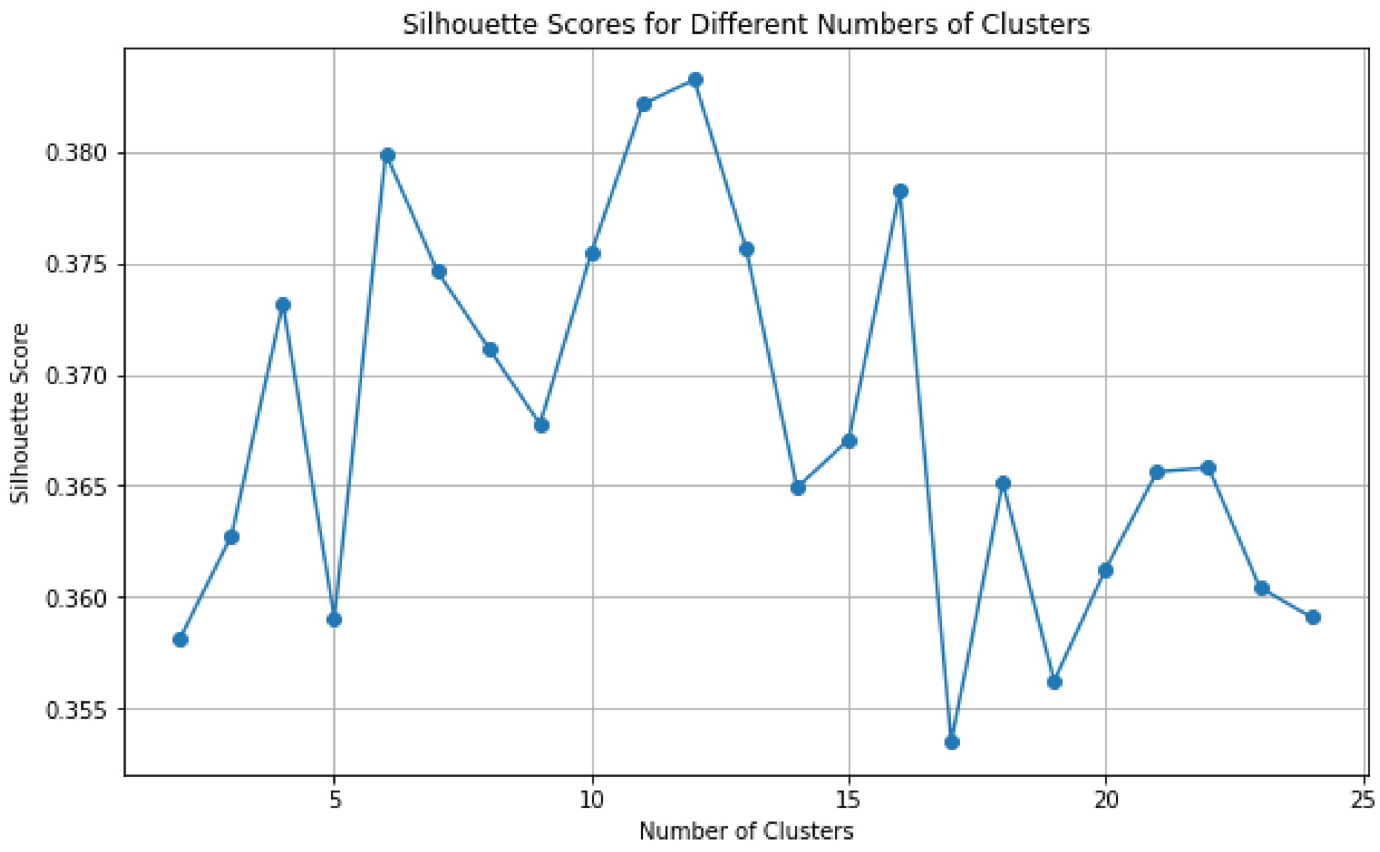
First of all: Set random. seed(14073479).

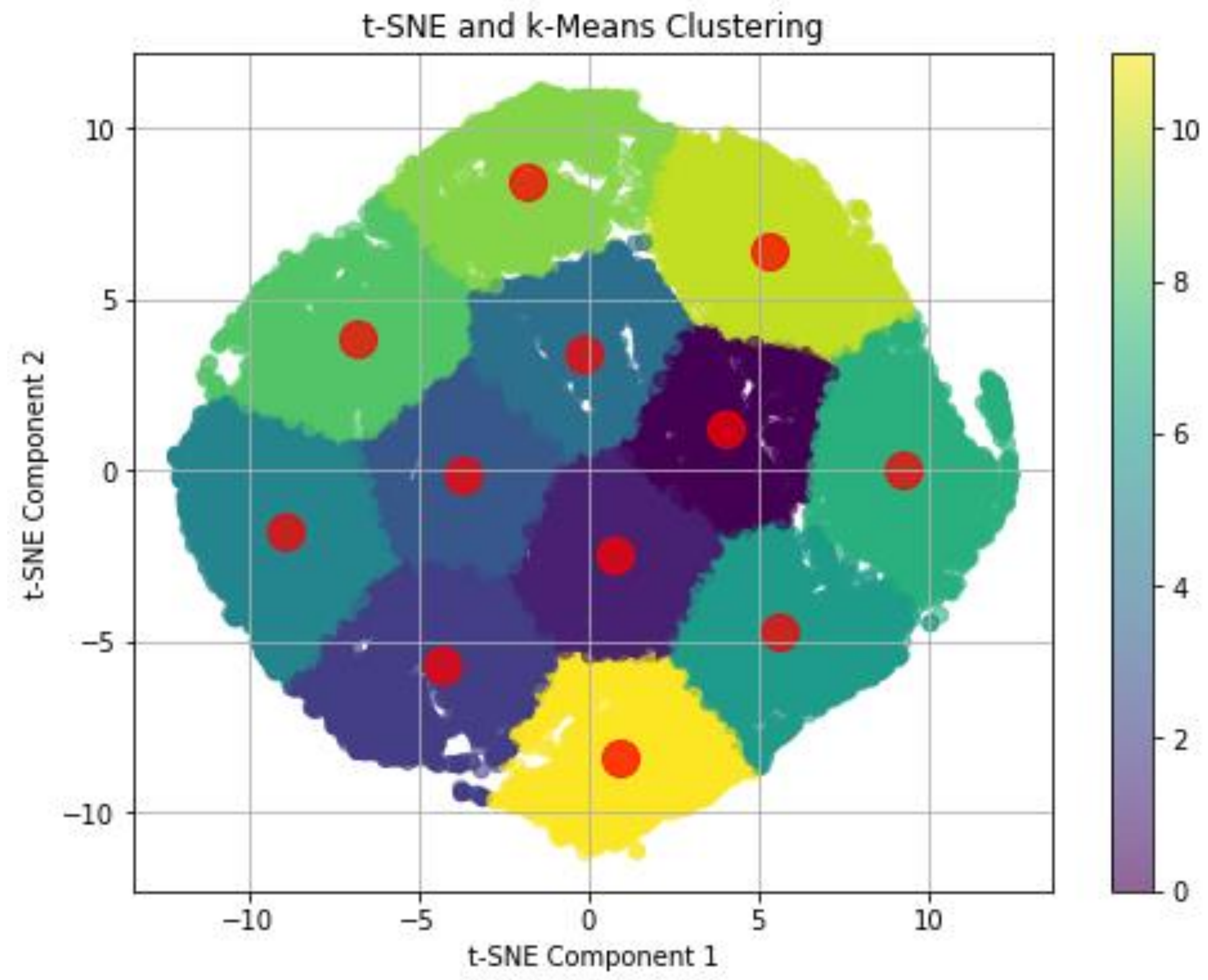
Data Processing: Main challenges and issues in the data management part: mode, genres and keys are in the string form, some missing data with ‘NaN’ or ‘?’ value exist in the dataset, train-test split: requires 500 songs from each genre to form a test set and the rest of the dataset will be the training set, I didn’t generate another validation set here because I choose to do the cross validation with K-Fold. First read data from csv file into a data frame. First drop all data with NaN or ? value involved, since there’re 10 genres and 5000 songs for each genre and the missing data is in a small amount. Then encode value inside the key column: create new columns each represent a single key and having 0 and 1 value to present the corresponding value of key for songs. And the mode column was imputed that major = 1 and minor = 0 to have a numerical value during the model training and testing process. Since there are ten different genres in the data set, I encode them with value from 1 to 10 for each genre. And the train-test split was made by selecting 500 songs for each genre as test set and rest serves as training set. The number of common ID between these two sets are checked to be 0 to ensure no leakage. And all numerical, non-categorial columns are normalized before any training and testing process.

Feature selection reason: The artists’ name will probably be a strong predictor when classifying the genre of a song. But more likely, it’s having correlation with the genre so that the model will have high chance of overfitting. And another serious consideration is that if new artist names are introduced in a new data set, how should the model evaluate the name? Thus, I drop the artists’ name and also the track name. And the reason I created new columns for each different keys is that I generate dummy variables for them as categorial value. Same for the genre column, every distinct value represents a genre. And since the dimensionality reduction’s result is not so ideal, I didn’t incorporate the result of TSNE in my model training procedure.

Dimensionality Reduction: Use T-SNE to reduce the dimensionality of the data set dropping four string columns: instance_id, artist_name, track_name and obtained_data. And the genre column is also excluded. All non-categorial data are normalized to eliminate the effect of data scaling. Then assess the TSNE result for number of clusters from 2 to 25 to find the optimal number of clusters that should be formed with highest Silhouette score. Finally plotted the clusters formed on the 2D display of data set using TSNE.

Dimensionality Observation: According to the Silhouette method of finding the optimal number of clusters that should be formed, there should be 12 clusters formed for this dataset. This is a pretty reasonable result: because even within a same music genre, there might be smaller classes exist like soft rock or hard rock etc. But the local structure of the data set is not very clearly displayed probably because: the acoustic features are not normally distributed which is an assumption made by TSNE method. And also, the data set mat not have a high-dimensional meaningful structure that can be captured in a lower-dimensional space. Because TSNE was designed based on the assumption of the existence of local structure and this structure is significant. Plots are displayed below. Total sum of distances of points to their cluster centers: 215369.875.





Model Choosing: I choose random forest model to perform the classification because tree-based method inherently classifies non-linearly separable data set and will probably have a good AUROC score with sufficient hyperparameter tunning. And other reasons why I choose random forest are: it highly reduces chance of overfitting to the noise, and efficient data usage with bootstrap. Also, since the acoustic features are unlikely to be normally distributed, I choose random forest model because it's robust which can diminish the effect of non-normal distributed data. And it's faster to train a random forest model with cross validation compared to other non-linear classification models. At the end, if the random forest model can't produce good performance, I will go for the gradient boosting and FNN. Actually, I prepare the code for gradient boosting without hyperparameter tunning and cross validation. As it produces similar ROC score as the random forest does, so I just focus on the random forest model because it's much faster.

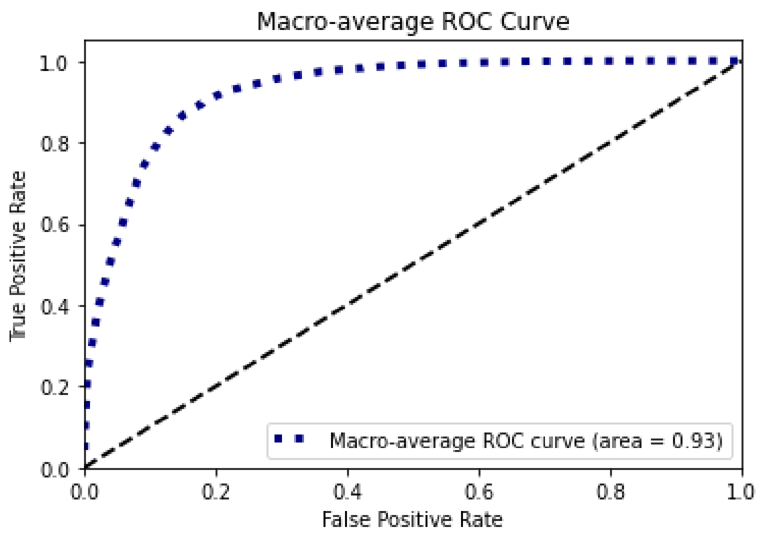
Model evaluation: Since this is a multi-class classification model, which means the traditionally used binary AUROC score will not be sufficient. So, I calculate the AUROC score for classifying each class and then take the macro average to get the final result. And I also use the probability estimates rather than the hard class predictions to be the result of the model. Because in a multi-class classification model, it's important to evaluate how well the model discriminate between each class. I also did the hard-class classification ROC curve just to make sure I didn't miss anything.

Hyperparameter tunning (grid search): Define a parameter grid which contains several different hyperparameter: max depth, min sample split, min sample leaf and bootstrap also their value that I want to test on. With 5 folds, there're 360 different fits from the grid search to find the optimal model which produces the highest AUROC score. Set the scoring to be AUROC score for the optimal result. And the value for each hyperparameter are discrete values like 3,5,10,15 instead of a range to save time for training.

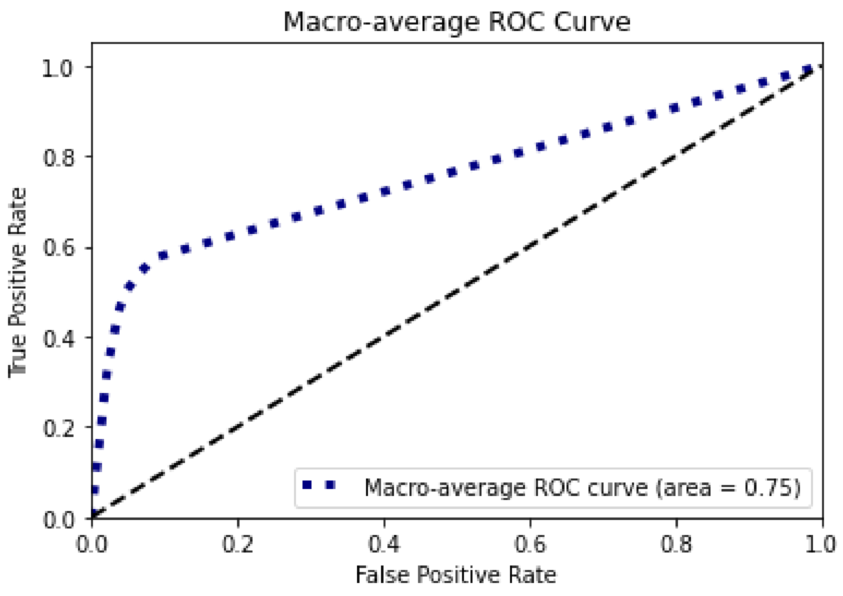
Cross Validation: Using K-Fold cross validation to improve the hyperparameter tunning. (Requires heavy computational resource) I personally set k=5 to reduce the time of finding the optimal model. The grid search library combines the feature of k-fold cross validation and hyperparameter tunning which makes it easier to find the optimal model with the scoring you want.

Model Performance: The final AUROC score calculated using probability matrix is 0.927 for the best model produced by cross validation which is good enough and plotted the ROC curve below. Individual class classification's ROC curves are also plotted but not displayed below to save some space. You can run the code to see the results. And the AUROC calculated by comparing the hard-class prediction dropped to 0.75 using random forest, which is also not bad. I also tried using the result of dimensionality reduction to train and test the model but only got 0.615 AUC score, so I didn't incorporate the result of dimensionality reduction in my model. The gradient boosting model also has 0.93 ROC score. But since random forest already produce a model with 0.927 AUROC score, we are just focusing on the random forest because it's much faster.

Roc produced by the probability matrix produced by X_test in Random Forest model



ROC produced by the exact class predicted by the X_test in Random Forest model



Conclusion: Random forest is a sufficient non-linear classification model to classify the music genre based on their acoustic features. And the most important factor for my model to success is the nature of tree-based method to inherently classify non-linear separable datasets. Just need some hyperparameter tunning and cross validation to make it optimal. And it’s fast to train, which is also important because this will allow me to test with multiple hyperparameters.