Preprocessing:

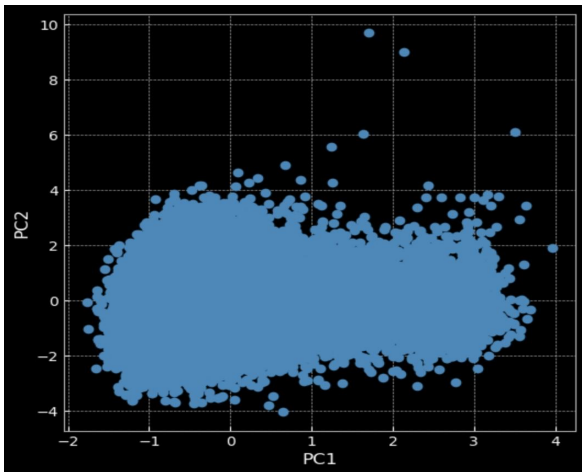
What I did first was deal with the missing data. I chose to drop the NA entries because there aren't many of them and after dropping them there are exactly 50,000 data left. I observed that in the "duration_ms" column there are 4939 values of -1.0, which doesn't make any sense since the duration can only be a positive number of seconds. Also, the entire "tempo" column is of string-formatted numbers and contains over 4980 "?" values. Thus, I converted the "tempo" column into a numeric valued column and did an imputation on those missing data so as to retain most of the information of the dataset. I replaced the missing values with the mean (or median) of the rest, with the method Dataframe.replace and Dataframe.fillna respectively. Then, I replaced the columns containing string information with numeric values. I replaced 12 different "keys" with numbers from 1 to 12, coded the "mode", Major and Minor, into 1 and 0, and transformed the category labels "music_genre" into the integers from 1 to 10. Lastly, I dropped the unrelated columns which I consider wouldn't help with the classification, including instance_id, artist_name, track_name, and obtained_date.

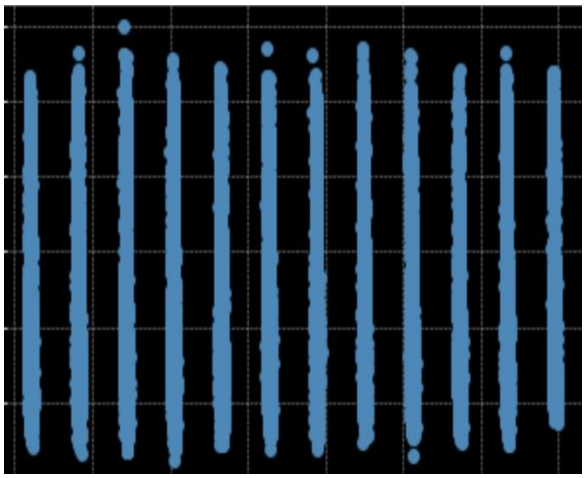
Train Test Split and Normalization:

I performed the train test set split before normalizing them to prevent any leakage. I used the "stratify" feature to obtain 5000 randomly picked genres (one per song, 500 from each genre) because each genre contains exactly 5000 songs. Then, I adopted Dataframe.apply(zscore) to normalize both the train set and the test set. Notice that I didn't exclude the categorical variables because after normalization they are still discrete numeric values that represent the categorical but with smaller space between each other. I also tried maintaining all the categorical values as instructed. The difference will be explained in the following sections.

Dimensionality Reduction and Clustering:

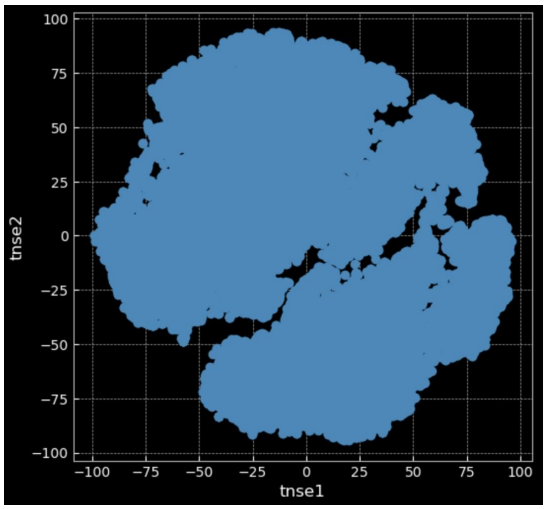
I first used PCA on the preprocessed training set, knowing that PCA can't deal with categorical data very well. Depending on the scale of the categorical value, if the categorical values are set large, like what I set from 1 to 12, the lower-dimensional plot will be completely dominated by this variable; however, if I represented the 12 keys with smaller numeric values, like normalizing them, in which case their relative distance is still retained, the plot would be drastically different. I tried building two models with both sets of labels and based on the AUC score I chose the latter one to perform the clustering and obtain the labels. By step-by-step analysis, the largest two eigenvalues account for 40% of the variance in the training set. I also performed the T-SNE and the multiple factor analysis FAMD(). Computing and comparing the silhouette score for each cluster, I chose the PCA and the optimal number of clusters given is 2. Another reason I chose PCA instead of other dimensionality reduction methods like MDS is that it is the fastest and easier to implement (MDS has a scary time complexity). Having chosen the cluster and the optimal k=2, I applied k-means to obtain the labels of the clustering result and appended them as an extra feature (column) in both X_train and X_test to boost my classification.

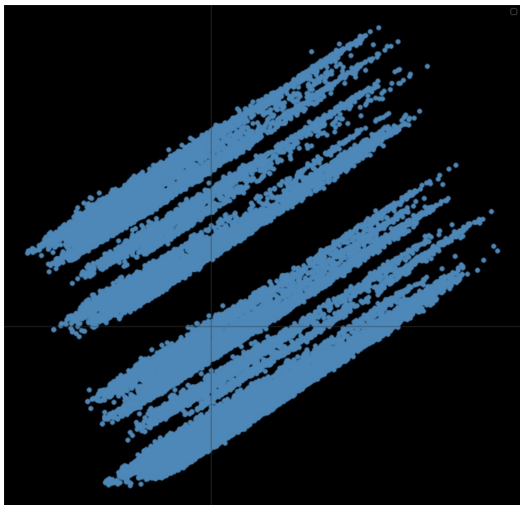




"Keys" represented by [0.1, 0.2, …, 1.2] "Keys" represented by [1, 2, …, 12] (not usable!)

(Different Plots of PCA depending on the numeric values of the categorical variables)





(Dimensional Reduction to 2D via T-SNE and FAMD)

For n_clusters = 2 The average pca silhouette_score is : 0.44490340
For n_clusters = 3 The average pca silhouette_score is : 0.41789929
For n_clusters = 4 The average pca silhouette_score is : 0.35585554
For n_clusters = 5 The average pca silhouette_score is : 0.36253052
For n_clusters = 6 The average pca silhouette_score is : 0.36061378

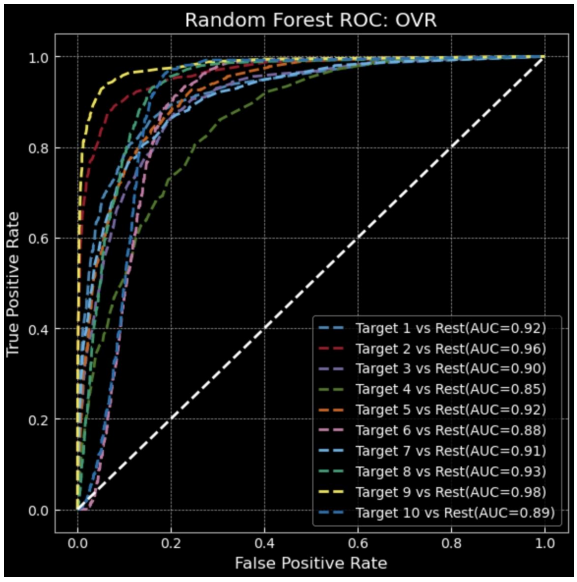
For n_clusters = 2 The average tsne silhouette_score is : 0.38887903
For n_clusters = 3 The average tsne silhouette_score is : 0.39409158
For n_clusters = 4 The average tsne silhouette_score is : 0.40149966
For n_clusters = 5 The average tsne silhouette_score is : 0.38612889
For n_clusters = 6 The average tsne silhouette_score is : 0.38980335

(Output from conducting the Silhouette method)

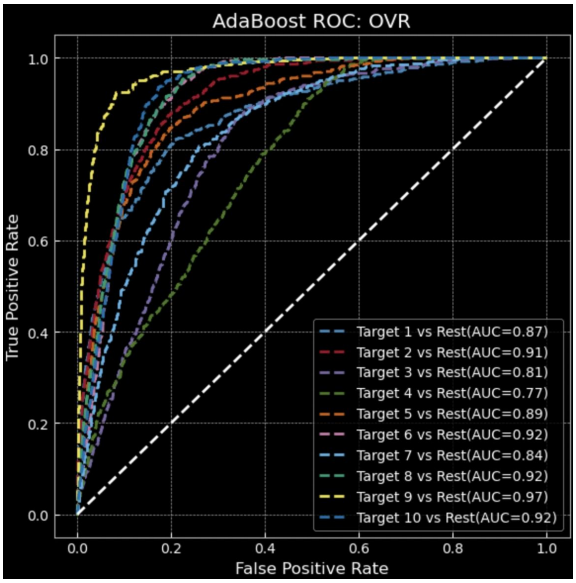
Classification:

I decided to feed the processed data to four different models to decide which one is best for the final classification. Respectively, I constructed a Random Forest, an AdaBoost, an SVM, and a Feedforward Neural Network. The AUC score for Random Forest is 0.91572575, a rather solid performance. The AUC score for AdaBoost is 0.8828176, slightly weaker than the previous one. The AUC score for SVM is 0.865556 and its accuracy is 50.3%. I consider the reason for it being the worst performance of all to be that the data is not linearly separable. The AUC score for Neural Network is 0.9054207 and its accuracy is 60%, which is another solid classification.

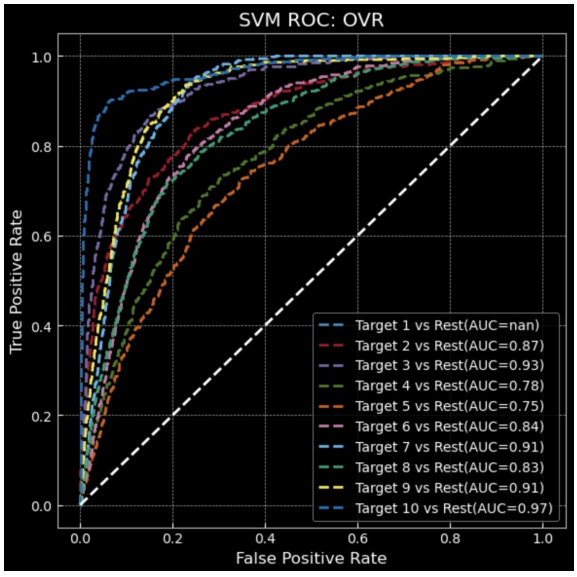
Based on the AUC score, I think the best model to use is the Random Forest. I have also plotted the One vs Rest ROC curves for all four models. The following ROC plots can confirm this conclusion since the curve for random forest has the nicest shape.



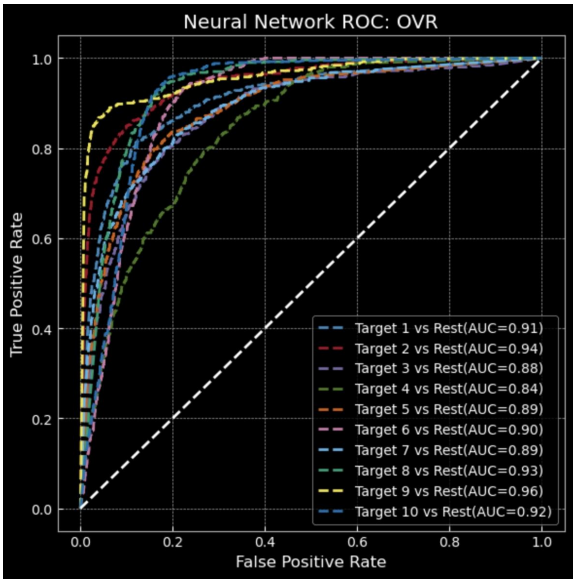
Random Forest ROC



AdaBoost ROC



SVM ROC



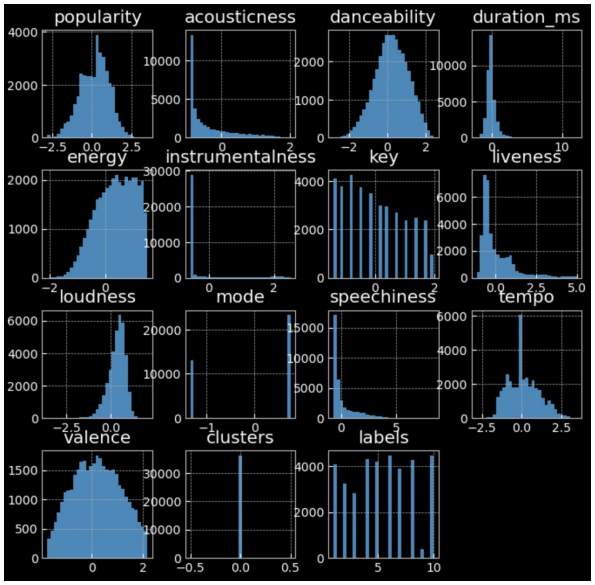
Two-Layered Neural Network ROC

To further improve the AUC score, I experimented with different preprocessing and clustering approaches. I discovered that a bigger n_estimators set in the RandomForestClassifier can improve the AUC score significantly. I compared different AUCs when the n_estimators is 100 and n_estimators is 1000, finding that the AUC increased from 0.91572575 to an amazing 0.91919722! The more trees in the forest, the better the classification result gets; nevertheless when the number of trees exceeds 1000, the improvement gets very little and might cause some overfitting. What's more, I adjusted the max_features multiple times and determined the optimal number to be 0.3, using which gives an AUC score of 0.9200337! I also tried different preprocessing methods. For example, I computed the model AUC when the acoustic features aren't normalized (also mode and key) and found the AUC to be 0.917 and even less. I also applied different clustering methods like T-SNE and found this wouldn't improve the model better than PCA. **Therefore, the highest AUC I can get is 0.9200337**.

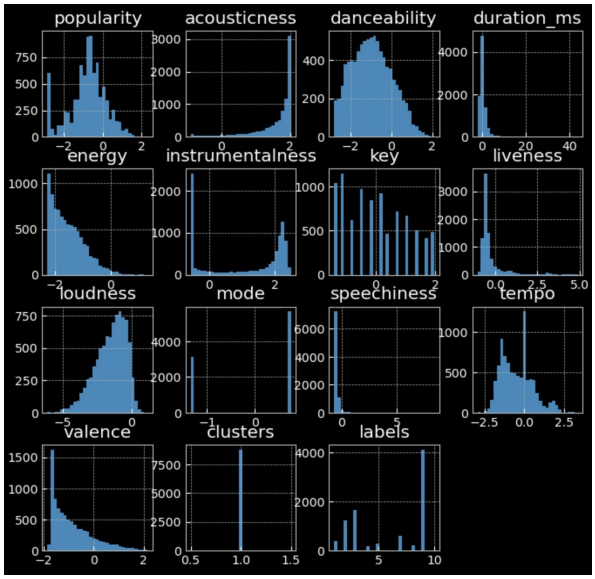
Extra Findings:

Other than the data visualization shown above, I also studied the influence of each factor. By observing the overall performance denoted by the AUC after dropping each factor, I concluded a group of effective factors. The more impact the predictor has on the random forest, the greater decrease it will cause to the AUC when it's absent. The effective factors include the popularity of the music, the danceability, the instrumentality, and the speechiness. Among them, the popularity of music seems to have the biggest impact. The rest three factors also generate a relatively bigger impact on the classification of my random forest model. In addition, by looking at the histograms of the two clusters generated by the PCA, I was able to see that the 'cluster' label can effectively

classify whether the song is classical or not, since under cluster 0 there's almost no "classical" songs when most of the "classical" songs are under cluster 1. The histograms are shown below:



Cluster 0 Histogram



Cluster 1 Histogram