Machine Learning Final Project—Music Classification

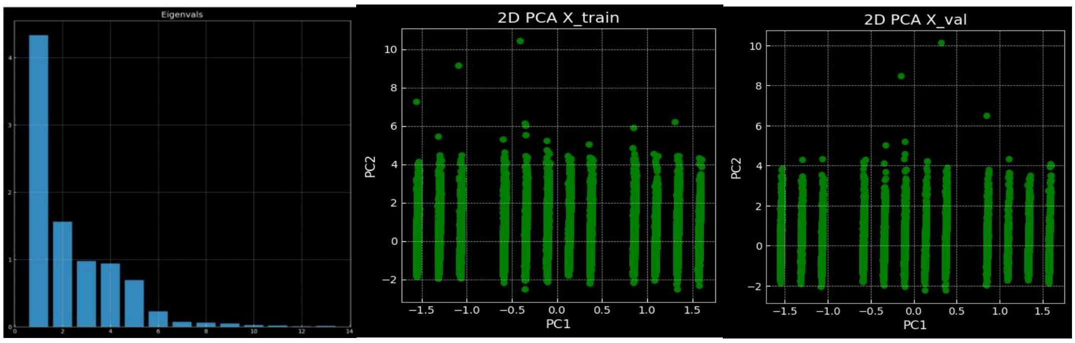
Zhiyang Ning zn2021

Cleaning Data: The first things we need to deal with is cleaning the data. We first perform dropna and only five rows are reduced. We also dropped the language columns, including song name, artist name, and obtained date (almost the same for all). Then, we transform the category labels to numeric labels, including key (to 0, 0.5, 1...6.5), mode (0, 1), and music genre (0~9). Through trying to plot the histograms of each column, we find that there's some problem with columns duration_ms and tempo, where the first one contains some "-1" and second one contains "?". So we replace them with the median of the column without these invalid rows. Another problem is that the dtype of tempo is str, so we transform it into float64 using a for loop. Up to now, we obtain a clean data and now perform the 8:2 train/test split. Training the logistic regression model with the current data, we have a **0.77** AUC and **0.30** accuracy. And we will see how normalization improve the result.

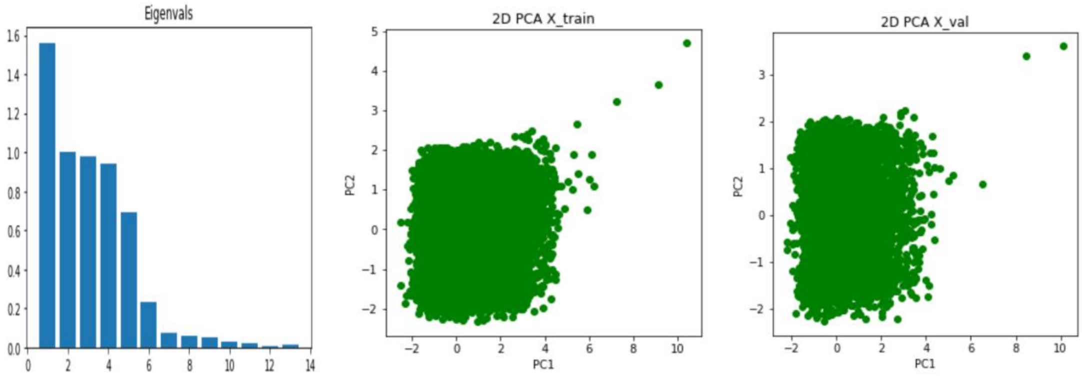
Normalization: We now perform scaling of the data—we normalize columns including popularity, durability, loudness, and tempo, so each of the column has roughly the same scale. But instance_id cannot be normalized and is with large scale—after trying with logistic regression and finding that instance_id reduces the model's performance greatly, we decided to drop this column for now. And training the logistic regression with the new data, we have a **0.886** AUC and **0.506** accuracy, which improves significantly compared to the above result.

Dimensionality Reduction: We then want to apply dimensionality reduction and clustering to find some hidden relations or clusters of the data. And we tried PCA and T-sne for the reduction process. We first compute the covariance matrix and eigenvalues of X_train:

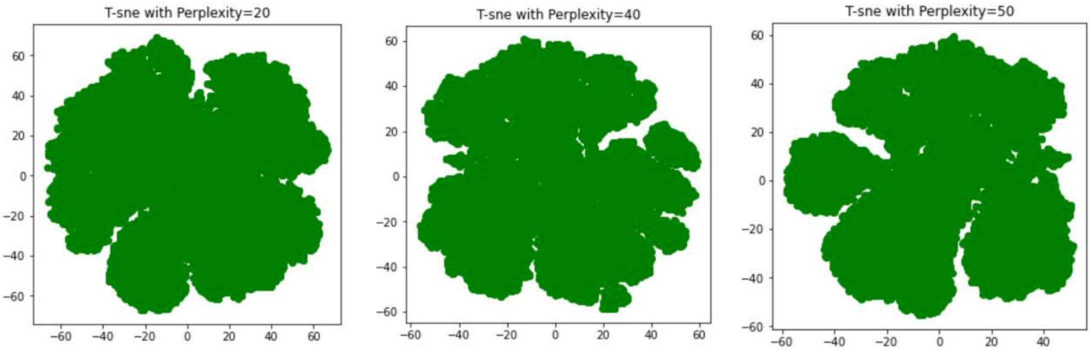
the largest two eigenvalues can explain **45.4%** of the total variance. We then independently (independent matrix) project the X_train and X_test on 2D space. The results are surprising,



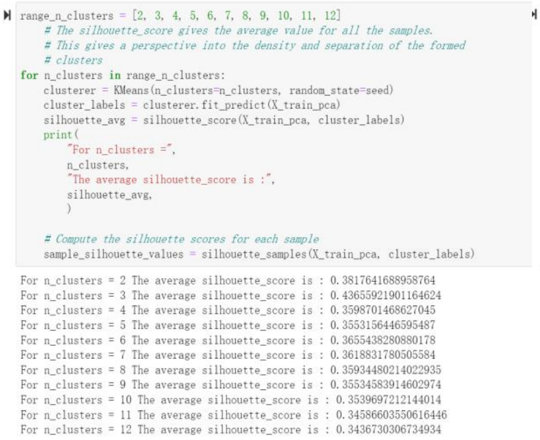
There are 12 vertical lines in the 2D plot, giving 12 perfect clusters, and I proceeded with this clustering to the end of project, where I suddenly realized that these clusters corresponds to the 12 key ranges—I didn't normalize these categorical variables so that their impact is too large. And this clustering is meaningless as it is a restatement of the key column. Then I came back to this part and normalized the key column. Repeating the process gives us these results.



We then perform T-sne and see what happens. And the results are as follows.



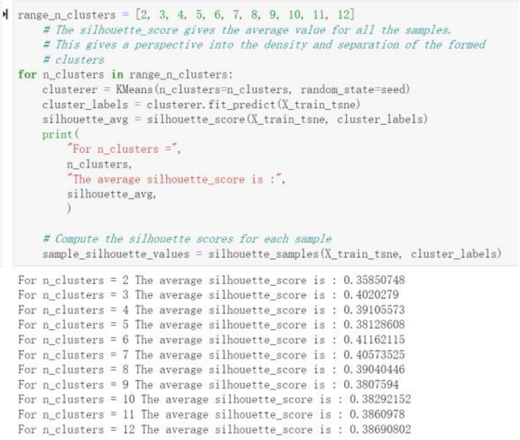
All of the six 2D solutions don’t provide clear clustering pattern. After we choose a perplexity of **40** for our T-sne solution to proceed on, we apply the Silhouette method to decide whether we should use PCA or T-sne and how many clusters are optimal.



```
range_n_clusters = [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
    # The silhouette_score gives the average value for all the samples.
    # This gives a perspective into the density and separation of the formed
    # clusters
for n_clusters in range_n_clusters:
    clusterer = KMeans(n_clusters=n_clusters, random_state=seed)
    cluster_labels = clusterer.fit_predict(X_train_pca)
    silhouette_avg = silhouette_score(X_train_pca, cluster_labels)
    print(
        "For n_clusters =",
        n_clusters,
        "The average silhouette_score is :",
        silhouette_avg,
        )

    # Compute the silhouette scores for each sample
    sample_silhouette_values = silhouette_samples(X_train_pca, cluster_labels)
```

```
For n_clusters = 2 The average silhouette_score is : 0.3817641688958764
For n_clusters = 3 The average silhouette_score is : 0.43655921901164624
For n_clusters = 4 The average silhouette_score is : 0.3598701468627045
For n_clusters = 5 The average silhouette_score is : 0.3553156446595487
For n_clusters = 6 The average silhouette_score is : 0.3655438280880178
For n_clusters = 7 The average silhouette_score is : 0.3618831780505584
For n_clusters = 8 The average silhouette_score is : 0.35934480214022935
For n_clusters = 9 The average silhouette_score is : 0.35534583914602974
For n_clusters = 10 The average silhouette_score is : 0.3539697212144014
For n_clusters = 11 The average silhouette_score is : 0.34586603550616446
For n_clusters = 12 The average silhouette_score is : 0.3436730306734934
```



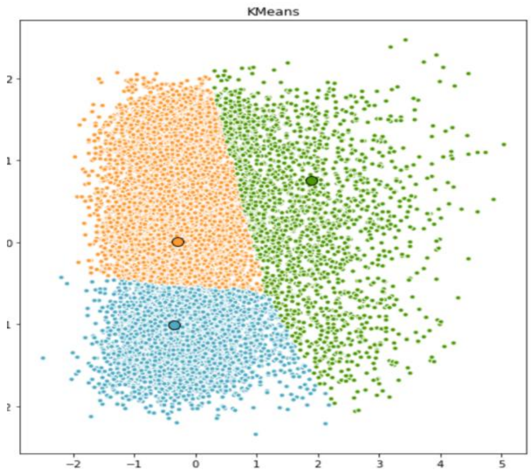
```
range_n_clusters = [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
    # The silhouette_score gives the average value for all the samples.
    # This gives a perspective into the density and separation of the formed
    # clusters
for n_clusters in range_n_clusters:
    clusterer = KMeans(n_clusters=n_clusters, random_state=seed)
    cluster_labels = clusterer.fit_predict(X_train_tsne)
    silhouette_avg = silhouette_score(X_train_tsne, cluster_labels)
    print(
        "For n_clusters =",
        n_clusters,
        "The average silhouette_score is :",
        silhouette_avg,
        )

    # Compute the silhouette scores for each sample
    sample_silhouette_values = silhouette_samples(X_train_tsne, cluster_labels)
```

```
For n_clusters = 2 The average silhouette_score is : 0.35850748
For n_clusters = 3 The average silhouette_score is : 0.4020279
For n_clusters = 4 The average silhouette_score is : 0.39105573
For n_clusters = 5 The average silhouette_score is : 0.38128608
For n_clusters = 6 The average silhouette_score is : 0.41162115
For n_clusters = 7 The average silhouette_score is : 0.40573525
For n_clusters = 8 The average silhouette_score is : 0.39040446
For n_clusters = 9 The average silhouette_score is : 0.3807594
For n_clusters = 10 The average silhouette_score is : 0.38292152
For n_clusters = 11 The average silhouette_score is : 0.3860978
For n_clusters = 12 The average silhouette_score is : 0.38690802
```

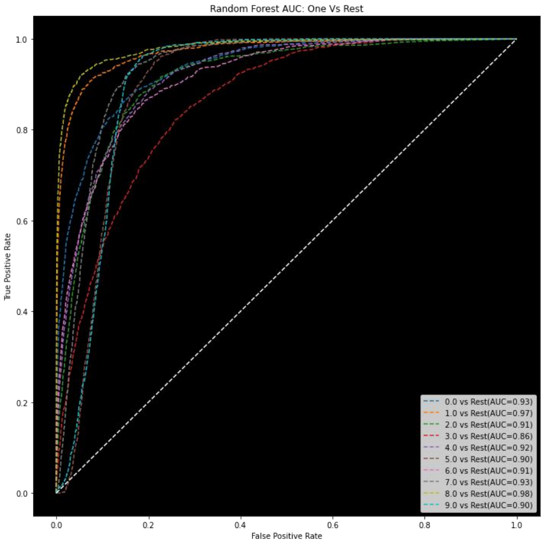
As we can see, the Silhouette score are quite close, the two choices are PCA with n=3 and T-sne with n=6. Using the labels provided from these two solutions, we do a logistic regression for classification merely using the labels. Labels from PCA with n=3 got an accuracy of **0.18**, which is quite good. While labels from T-sne with n=6 obtained an accuracy of **0.07**—worse than random guessing. So we decided to use the PCA solution.

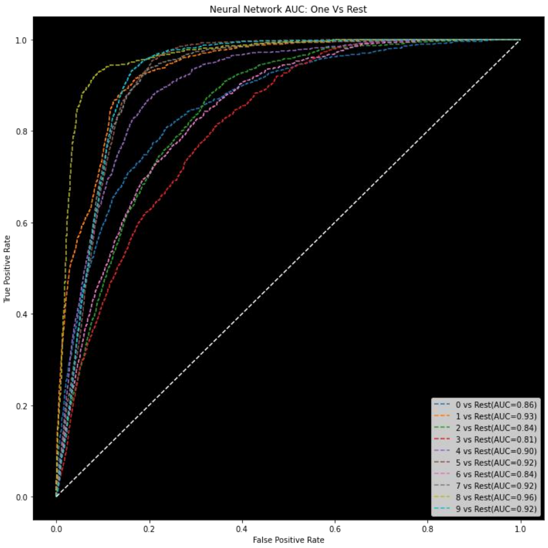
Clustering: We applied n=3 for clustering. We fit the Kmeans with the X_train_pca data, and used it to predict the labels of X_test_pca. Then we take the labels and append it to our original X_train and X_test to form a new feature. Here’s the X_train_pca clustering.



We will then try with different models on the two sets of data—1) original data without cluster feature; 2) data with cluster feature. We will compare and interpret them and in order to find out the best classifier.

Classification: First, we want to examine the effectiveness of the clustering feature, so we applied logistic regression one the single clustering column to fit the genre. The AUC is **0.599**, and the accuracy is **0.188**, which is better than random-guessing and shows that the clustering feature might be useful. Second, we apply logistic regression again to X_train with clusters, and we obtain a **0.887** AUC and a **0.509** accuracy, which is slightly better compared to the previous logistic regression. After trying the relatively simple logistic regression, we implement some more complicated models. We tried SVM, Random Forest, and Adaboost each on type 1) and 2) data mentioned above. While SVM and Adaboost did not yield significant result, Random Forest achieved a **0.922** AUC and a **0.550** accuracy for data1) and a **0.551** accuracy for data2) We then try with the multiple layer neural network for multiclass classification. After converting our data to tensor, we create a two-hidden-layer neural network, with 12 nodes for each hidden-layer and ReLU() being the two activation functions. The result for the neural network is even better than the random forest: an accuracy of **0.568** for data without clustering feature, and **0.5694** for data with the clustering feature. The AUC is hard to determine since we have negative and large values in the output that cannot be used as probabilities for calculating the auc(). And if we add Sigmoid as an activation function in NN, the performance drops significantly. Therefore we directly apply sigmoid() to the output and divide each value by the sum of its row to obtain the "0~1 probabilities." The resulting AUC is **0.869** and **0.902** respectively, which should be used for reference only since the calculation method is different from the AUCs before. The higher accuracy suggests that **NN** might be a better classifier in this case. The following plots shows the 10 “one vs rest” ROC curves for Random Forest and Neural Network. (Notice that steps for calculating AUC differ for the two models)

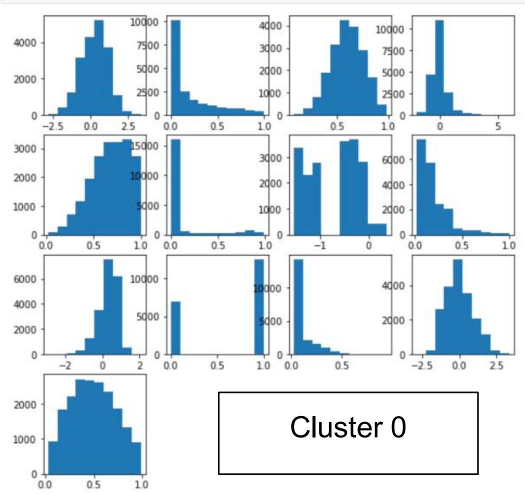


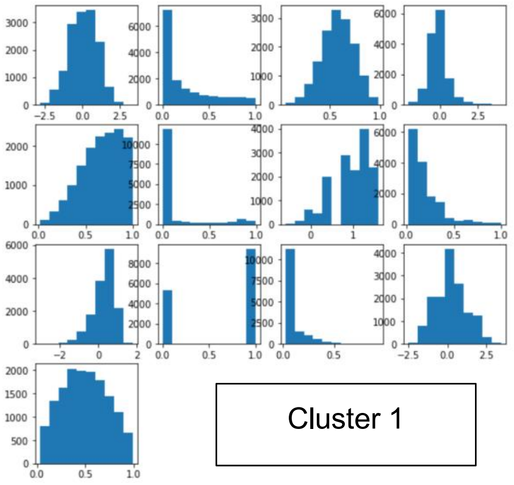


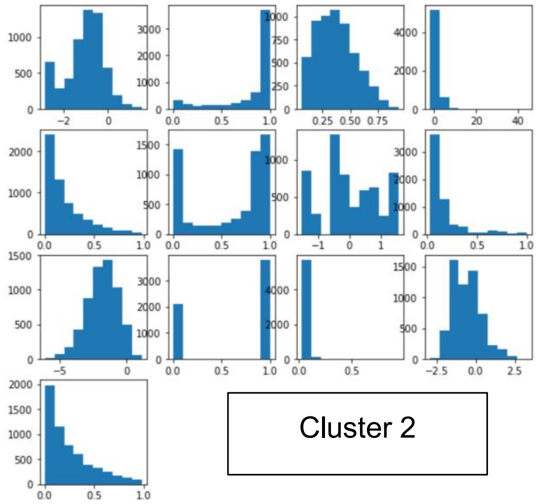
Conclusion: Through multiple cross comparisons, we can conclude that normalization, dimensionality reduction, and clustering can help booster the performance of classifiers. For the best performance in classification purpose, we should choose between the two complicated models—Random Forest and Neural Network—as Random Forest provides an AUC of **0.922** and Neural Network gives an accuracy of **0.569** (ideally its AUC should be more than 0.93 if we can apply the same calculation method used for Random Forest). Given these statistics and comparisons, I would prefer neural network for this classification task.

Extra exploration are at the next page :)

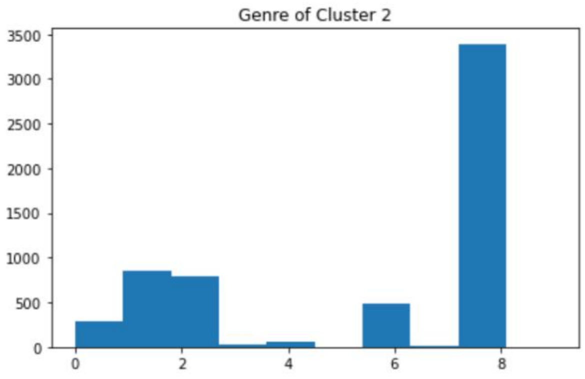
Exploration of Clusters: We can locate the three clusters by their labels and plot the histogram with their members' columns.







We can observe that the major differences between cluster 2 versus 0 and 1 are the first, second, fifth, and the thirteenth columns, which correspond to low popularity, high acousticness, low energy, and low valence. And these features should represents music genre of Jazz and Classical. And we can take a look at the true genre for verification—we can see that Classical Music takes a huge proportion in this cluster. Looking back to our neural network ROC plot, we can find out that the Classical AUC is the highest among 10 genres, reaching 0.96>0.95 of original data AUC, which further demonstrate dimensionality reduction and clustering procedures might help discover hidden relationships within data and improve the classification


Genre of Cluster 2