Data Preprocessing

Before anything, I use the .isnull() function to check the number of NAN values for each column, which is the most straightforward way to look for missing values. Suspiciously, there are exactly five missing values for each column. I pull them out and find out that this dataset contains five blank rows, therefore I just drop them, making the total number of rows 50000, which is correct. Then, I look at the csv file and notice that for different columns missing values present in different ways. For duration, the missing value is -1; for instrumentalness, the missing value is 0; for tempo, the missing value is "?". I use mean imputation for these columns: replacing characters stands for missing value by the mean of the columns.

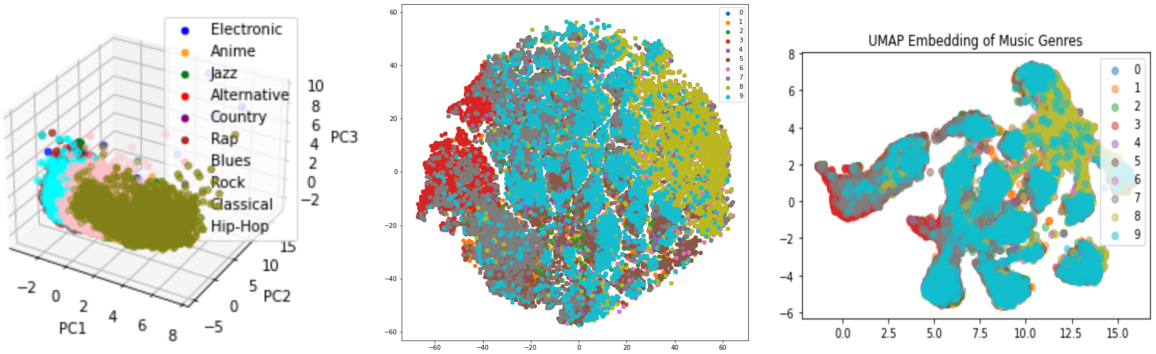
There are two columns that contain categorical data: key and mode. I use map function to transfer twelve different keys from categorical string data into numerical data from 1 to 12. For mode, I transform Major to 1 and Minor to 0. I one-hot encoded the 'key' data into twelve dummy variables. The target value, genre, is also categorical string data. I use LabelEncoder() to transform them into numerical data (0 to 9) to fit the classification model. Then, I drop the columns containing data that are irrelevant to the classification: Spotify ID, artist name, song name, and obtained date.

For the train-test splitting the data, I do as the spec sheet says: for each genre randomly select 500 songs as test set and use the rest as training set, with my ID as the random seed. To avoid leakage, I use the drop function to separate two sets. Then, I define X and y for the train and test set. And that's all about how I preprocess my data.

Classification and Dimension Reduction:

I understand that the spec sheet requires us to do dimension reduction before classification, and I couldn't agree more on this approach, since using all the data to build and train a model can take a tremendous amount of time, especially for SVM or neural networks. However, I want to have a general perception on how different models originally perform on this dataset, so I built three different models (SVM, adaBoost, and Random Forest) for better tradeoff decisions. The result shows that, even though they all perform well with AUC score around 0.9 for each genre classification, adaBoost does not perform as well as Random Forest, and SVM performs in a peculiar way, which I will discuss in Struggles & Difficulties section, and takes horrendous amount of time to run. Therefore, I pick Random Forest as the model to reach for the optimal AUC score.

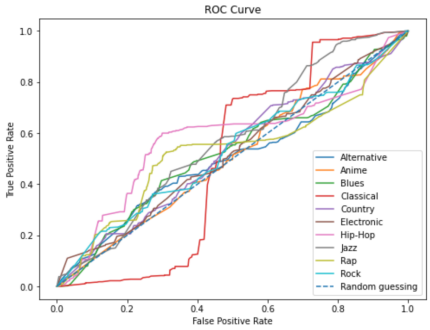
I tried three different dimension reduction methods: PCA, t-SNE, and UMAP. As the spec sheet required, I did not one-hot encode the categorical data, and the plots are below:



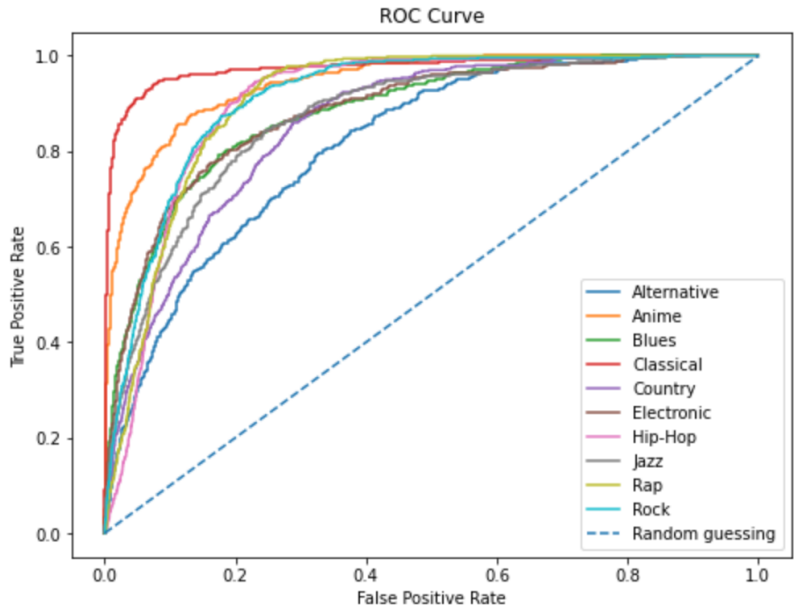
Based on the plots, it seems that PCA returns (a little bit too) clear clusters of the data, while t-SNE and UMAP have the data all overlapped when reducing it to a lower dimension. Therefore, I prefer PCA for getting the optimal result, and it is confirmed in the later process.

Getting the (local) Maximum AUC & Key to Success

Since the spec sheet asked, I then tried the three classification models after dimensionally reducing the data first. I first used PCA to transform the data, choosing components with an eigenvalue larger than 1 (n=3), built three different classification models with the transformed train data, and then fitting the transformed test data to check models' performance by returning the AUC score for classifying each genre. The result is the same as using the original data: Random Forest model returns the most desirable outcome, with the highest overall AUC score for all genres. Therefore, I again confirm my choice of the model. To see if other dimension reduction will return a better outcome, I tried data transformed by t-SNE and UMAP to train and test the model. UMAP returns a reasonably good outcome, even though not as good as PCA, and t-SNE returns an unacceptable outcome. Unlike UMAP, the structure of t-SNE is uninterpretable, and it will lose too much information for good classification. To test whether t-SNE transformed data would perform better in other classification methods, I built an adaBoost model with t-SNE transformed data, and the result was even worse.



At this point, I settled with PCA and Random Forest as the ultimate method, and started to aim for a better AUC score for each genre. I first checked if I can improve the dimension reduction process, and found out that even though I chose all four components with eigenvalues higher than 1, it did not explain enough of the total variance. Therefore, I chose components that could explain more than 0.8 of the total variance. It indeed to a certain extent increased the general AUC score. Then, I aimed at hyperparameter tuning. When I was testing adaBoost and tried to find an optimal result, I used GridSearch for the best hyperparameter which led to the optimal AUC score, and the model's performance increased about 0.02 AUC score for each genre. For the Random Forest model, I also intended to do GridSearch, but due to the gross incompetence of my computer, it could not be run. I resigned to use RandomSearch instead, in hope of finding a good hyperparameter while not exhaustively searching for all possible hyperparameters in a reasonable range, but I had to declare failure after two days and nights of running and overheating and multiple shutdowns of the computer. Finally, I compromise to manually pick reasonable hyperparameters and run the model multiple times for the best result. Still, I managed to get a desirable outcome with the best AUC score I get out of all combinations of dimension reduction methods and classification methods, even better than the adaBoost with the best hyperparameter.




```
AUC scores:
Alternative: 0.813
Anime: 0.945
Blues: 0.883
Classical: 0.975
Country: 0.856
Electronic: 0.882
Hip-Hop: 0.904
Jazz: 0.870
Rap: 0.907
Rock: 0.912
```

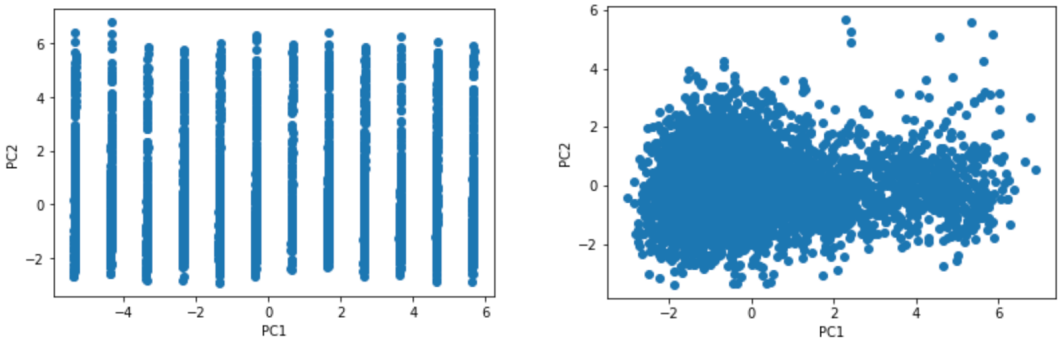
There are several factors that are essential to a better classification. As I've discussed, choosing the right dimension reduction method and classification algorithm for the nature of the dataset is important. Hyperparameter tuning also increases the model performance. Preprocessing the data carefully is required to avoid scaling problems. But for me, in this project, my biggest take away, and frankly the fundamental key to success in classification, is how to build the model properly. It might sound simple and straightforward, but it is actually essential for this project. The classification goal of this project is different from all the previous assignments we have encountered, which were all binary classification. However, this dataset required us to make 10 clusterings, which led to the fundamental difference in building the model with previous assignments. The model needs to be built and trained for each genre; that line of code needs to stay inside the for loop, and this is my biggest lesson.

Struggles and Difficulties

Besides all the scaling or datatype problems which I elaborate on in the Data Preprocessing part, one of the biggest difficulties is to make ten classifications, which is different from all the previous binary classification exercises. I started up, unfortunately, with SVM, and it performed so well with the model being built outside the loop for all genres. I used the same approach with adaBoost and Random Forest which led to the most horrible result: the accuracy score for each genre was reasonably high for about 0.8, but the AUC score could be as low as 0.1: the false positives prevailed, and I imagined this was what the spec sheet said about more way to be wrong than be right, and it showed the importance of looking into AUC score instead of just accuracy. After I found out the more correct way, which is to build and train the

model for each genre, SVM became impossible to run without dimension reduction, which leads to the second main theme of the struggles. Even though the data set is not too large, way smaller than the regression data set, it is still quite a bit to ask for the computer.

Something Trivial, Something New



The difference between the two PCA result, is caused by whether one-hot encoding the 'key' variable or not.