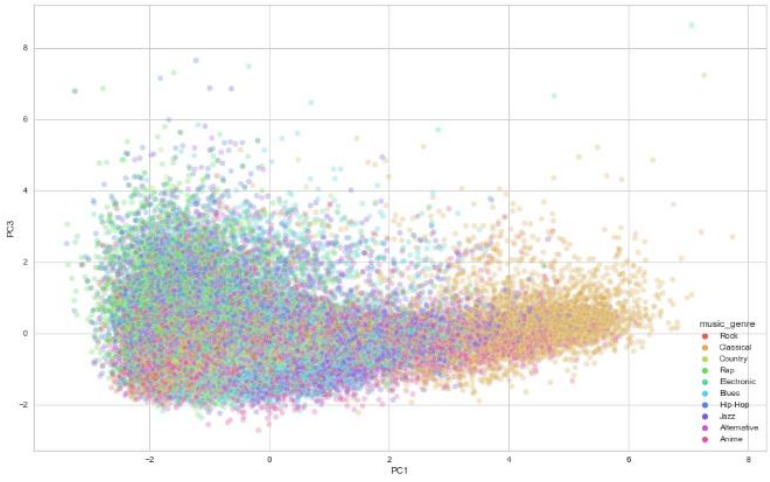
Justin Chang
jdc690

Final Report

The first step in building my model was to clean the data of nans and unnecessary values. There were three columns that were dropped: ‘instance_id’, ‘track_name’, and ‘artist_name’. I chose to drop these columns because they did not seem to be helpful in predictions. There were too many unique values in these columns to be helpful. While artist name and track name could be useful if the text was examined, instance_id is just an identification number and not helpful towards predictions. The features ‘tempo’ and ‘duration_ms’ including values of ‘?’ and -1 respectively. Before dealing with these values, I did a train test split on the data to avoid any leakage problems. To keep the same distribution of target labels after splitting (as specified in the spec sheet), stratify was used. Changes to the data after the split were made both on the test and training set. After the split, both datasets were imputed with the median of their column as a simple solution. There were also several features that had their data type changed such as popularity to int, duration_ms to int, and tempo to float.

The next step after data cleaning was preprocessing. Looking through the histograms of the numerical data, it was clear that the features acousticness and instrumentalness were both heavily skewed and in need of normalization. After testing out several different normalization techniques, a simple square root transformation did the best job of reduction the skewness, so it was applied to the data. There were 2 categorical features in the data as well, mode and key. These features were one hot encoded with the pandas get_dummies function. Standard scaling was also important, otherwise the PCA would not perform as well, so sklearn’s StandardScaler was used on the continuous features of the data. PCA and t-SNE were performed on the data. Based on the eigenvalues in the scree plot, I used the first 3 principal components based on the kaiser criteria. Looking through the plots for PCA and t-SNE, the PCA seemed to be much better at clustering. I also understood what the principal components meant through interpretations. The dummy variables were separated during the Standardization and dimension reduction process and added back in after both processes. This is because dummy variables are not meant to be scaled or put through PCA.



Looking at the clustering on the left, we can see that while there are some distinct groups, many still overlap each other. More angles of the clustering are in the notebook.

The third step I did was briefly testing various models and figuring out which one would work best for the classification process. Various models were tested using both untouched data and the PCA data such as LogisticRegression, AdaBoost, SGDClassifier, and Random Forest.

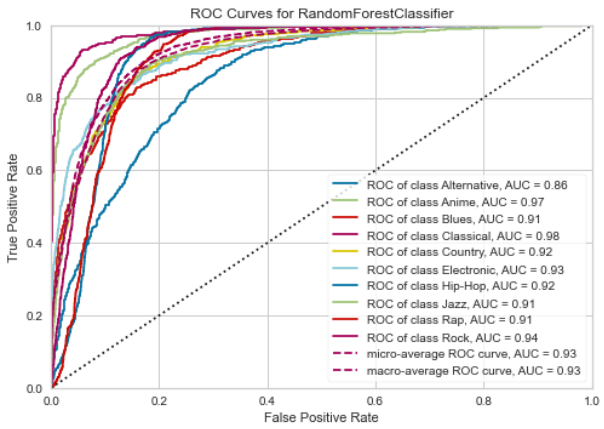
Justin Chang
jdc690

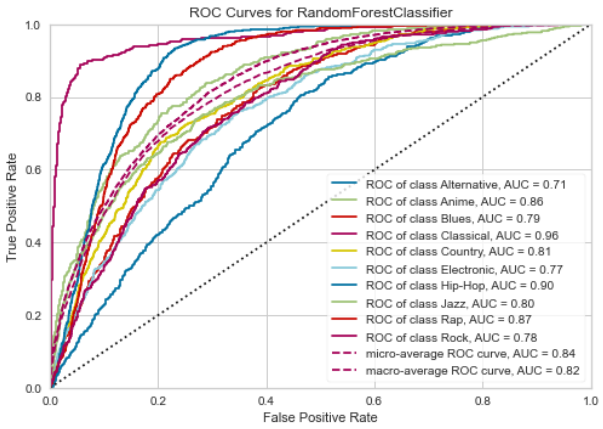
These models used a train test split within the training data. I did a brief evaluation of each model by looking at the avg AUC for each class with 'one vs rest' classification as well as accuracy. Looking at the metrics, it seemed that Random Forest had both highest AUC and accuracy in both the non-PCA and PCA reduced data. I ended up choosing Random Forest as my model for both dimension reduced and untouched data. Interestingly, I also found that my non-numerical target labels worked fine in the classification models and there was no need to transform the labels into numerical ones.

The final task was to parameter tune the random forest model so that it had the best possible performance. I did this by combining 2 different methodologies and using both to figure out the optimal hyperparameter. The first method was to iterate through each hyperparameter individually one by one and figure out what was the best value for the specific hyperparameter independently through plots. The second method was to do a grid search on the hyperparameters using sklearn's ReandomizedSearchCV algorithm to try a variety of random combinations with the hyperparameters as well as cross validate. After looking through the results of both methods, I found the hyperparameters that worked best. Hyperparameter values can be found in the notebook.

With my final model I was able to get an AUC score of 0.9250455555555556 on the model that used the non-reduced data. The model that used the PCA data managed to get a AUC of 0.8241288000000001. This is not bad considering the fact that there are 10 classes, but it is clear that the model without PCA performs better. This is also shown by the plots below.

Non-PCA Random Forest on the left, PCA Random Forest on the right.

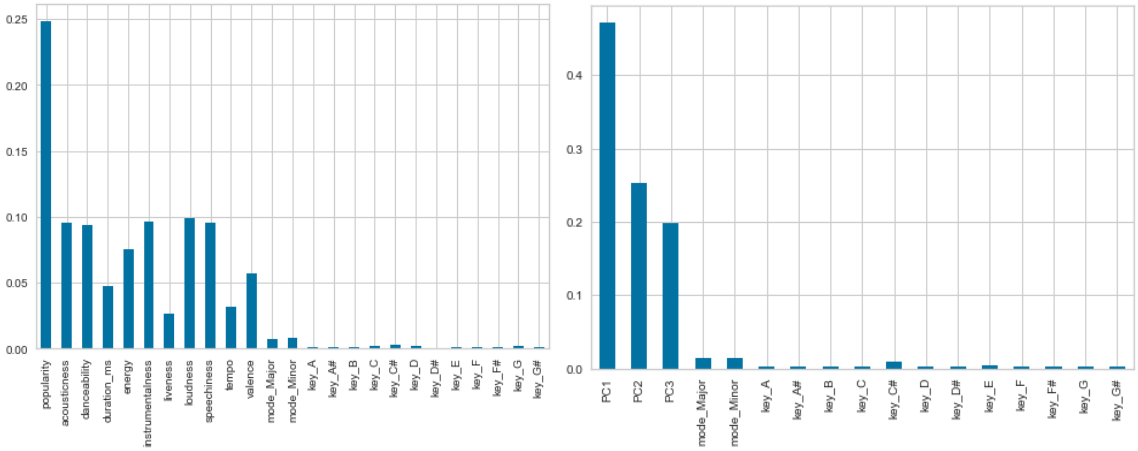




I believe that dimension reduction is unnecessary here for several reasons. Many of the features are not correlated at all or show very little correlation among themselves. Additionally, by doing the dimension reduction, information is being lost and the model may not be able to predict as well. This is evident in testing the models and the final model, where the models fit on the non-reduced data consistently outperformed the PCA data models. If I were to choose one model out of my 2 final ones, it would be the non-PCA Random Forest.

Justin Chang
jdc690

Non-PCA Random Forest on the left, PCA Random Forest on the right.



Based on the bar charts above, we can see that for the non-PCA model, the most important feature in the model is the popularity of the song, while for the PCA model, PC1 is the most important.