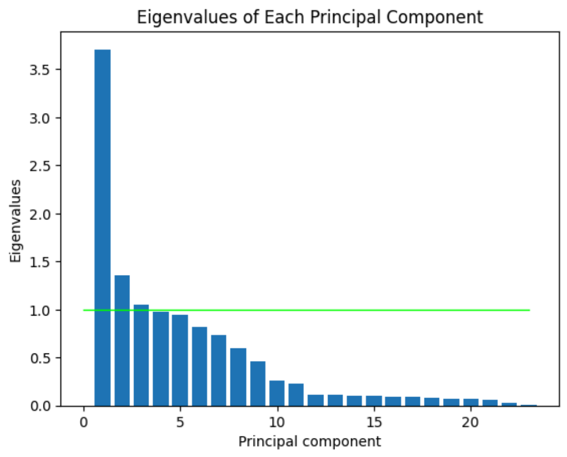
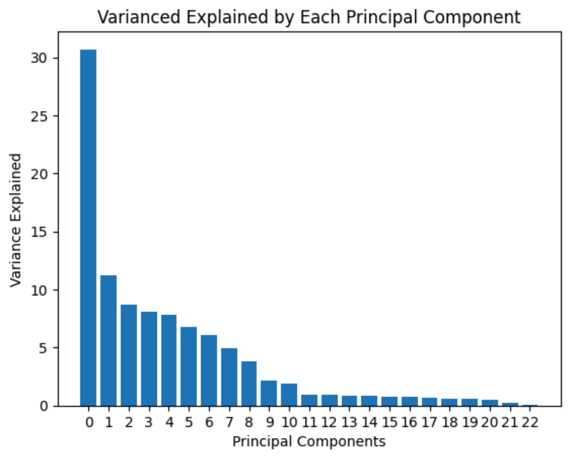
Jack Lee

Song Genre Classification

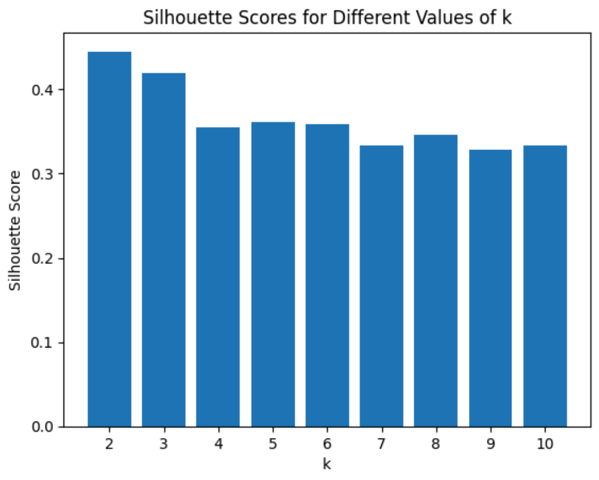
The dataset used for this project is 50,000 randomly picked songs from an API released by Spotify, which includes audio features of each song. Because this is a real dataset, there were many questionable values within the dataset that needed to be fixed before building any sort of model. The first task was to deal with NaN values within the dataset. Originally, the plan was to impute the missing data using the median of each column (assuming that column had numerical data) of each genre or the mode if the column was categorical. However, whenever there were any missing values, the entire row (or all the features of a song) was missing. Thus, because there were only 5 rows with missing data and imputing these rows without knowing anything about the song would be as good as random guessing, the rows with missing values were simply dropped. The next task was to deal with columns with category labels, which were the ‘mode’ and ‘key’ columns. The ‘mode’ column only had two categories (Major or Minor), so rows where the ‘mode’ column was equal to ‘Minor’ were replaced with 0 and 1 for ‘Major’. On the other hand, the ‘key’ column had 12 categories ('A#', 'D', 'G#', 'C#', 'F#', 'B', 'G', 'F', 'A', 'C', 'E', 'D#'). Therefore, the ‘key’ column was dummy coded using pandas ‘get_dummies’ function, which created new columns named ‘key_1’ through ‘key_11’ with binary values indicating whether each song corresponds to the respective key. ‘key_0’ was not needed because if a row had a value of 0 for columns ‘key_1’ through ‘key_11’, then that means that song is in key 0 (A# in this case). The original ‘key’ column was then dropped. The ‘instance_id’, 'artist_name', 'track_name', and 'obtained_date' columns were also dropped as there are too many unique values in these columns to reasonably dummy code it into labels or to give any helpful information in classifying the genre of the song. Finally, there were also instances of dirty data within the dataset. For instance, in the ‘tempo’ column, some rows had “?” as its value and in the ‘duration_ms’ column, some rows had -1 as its value, which are clearly incorrect. Instead of dropping these rows, they were imputed using the median of the column for each genre. However, before imputing, the dataset was split into training (4500 random songs from each genre for a total of 45000 rows in the training set) and test (500 random songs from each genre for a total of 5000 rows in the test set) sets first. Then, using just the training set, the median tempo and duration of a song of each genre was computed. These values then replaced the dirty data in the training set. Now, instead of finding the median of the test set, the median computed in the training set was imputed into the test set for each respective genre. In other words, if the median tempo of a hip-hop song was 100 in the training set, the dirty data in the tempo column of the test set was also replaced with 100, assuming the genre of the song was hip-hop. This is because the test set plays the role of fresh unseen data, meaning the test set should not be accessible during the training stage. Imputing the dirty data in the test set using the median of the test set would be using information in the test set when the model is run on the test set. This causes data leakage, which also introduces potential bias in the performance of the model. A similar methodology was used for normalization. The StandardScaler object was fit to the training set and then the test set was transformed using the scaling parameters learned from the training set. This ensures that the test data is scaled using the same mean and standard deviation as the training data, which is important because using a different scaling for the test data could introduce bias and affect the performance of the model due to data leakage for the same reasons mentioned during the imputing stage.

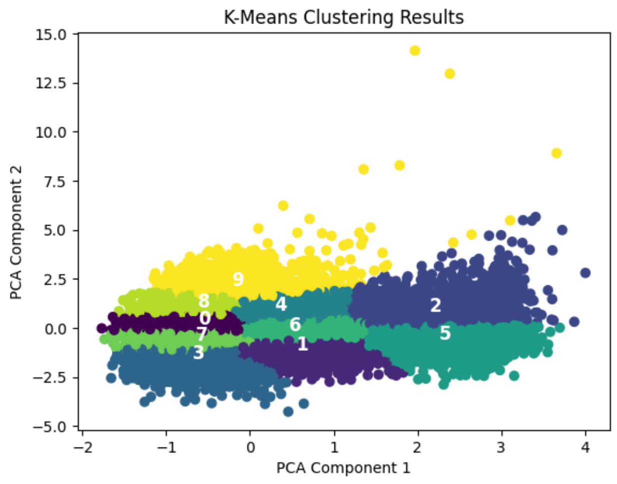
Next, the features and target variables were separated for both the training and test sets and PCA was done to the training set without the target (genre) variable. These are the graphs for the eigenvalues and variance explained by each principal component:



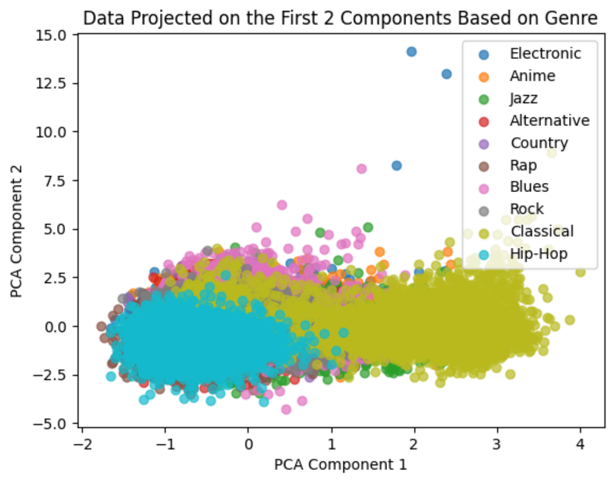


But to first visualize the data in a 2D space, the training set was reduced using a PCA with 2 components. However, the variance explained by only the first two components is less than 50% (~42%). Therefore, when KMeans clustering was performed on the reduced data, it did not do very well in differentiating the songs:



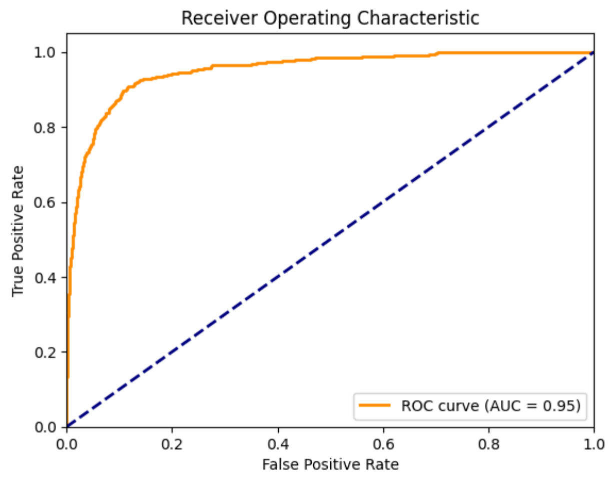


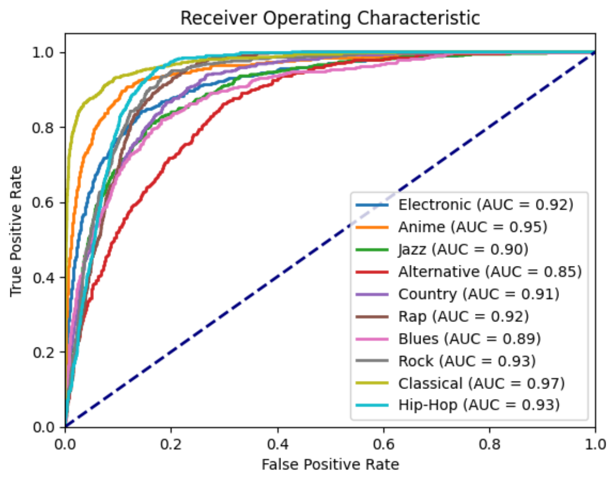
Because there are 10 different genres that the songs can be classified into, the optimal number of clusters should be 10. However, as shown in the bar plot, the silhouette score is quite low when k=10. The cluster visualization tells a similar story, as it looks like one blob of data without any clear way to classify one cluster from another.



The graph above explains why the clustering is poor. With only two components, the songs are nearly indistinguishable from one another and it is impossible to classify them into different genres. Hence, it would not be a good idea to fit a classifier with a PCA transformed dataset using only 2 principal components. Therefore, the classifier used a dataset that was transformed using PCA with 8 components, as the first 8 components all had eigenvalues above 0.5. Using 8 components explains most of the variance (~84%) while keeping the dimensionality relatively low.

Support vector machine (SVM) was the classifier used to predict the genre of a song. This classifier was chosen because it can handle non-linearly separable classes, which is important since as shown in the data projection plot, the songs are clearly not separable. In addition, SVMs work well in high-dimensional spaces, which is also important as with this dataset, it is impossible to build a model with only 2 components. Therefore, since the classifier has to work with a higher dimensional dataset, it is important to choose a classifier that can deal with high-dimensionality. After training the model using the training set, running it on the test set resulted in an overall AUC of 0.95 and the AUC scores of each individual genre is shown on the right.

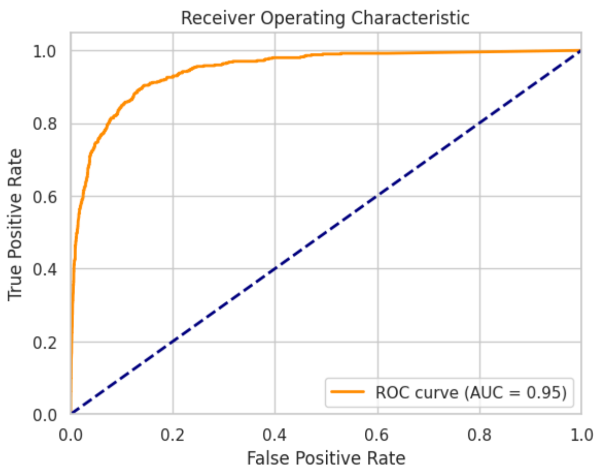


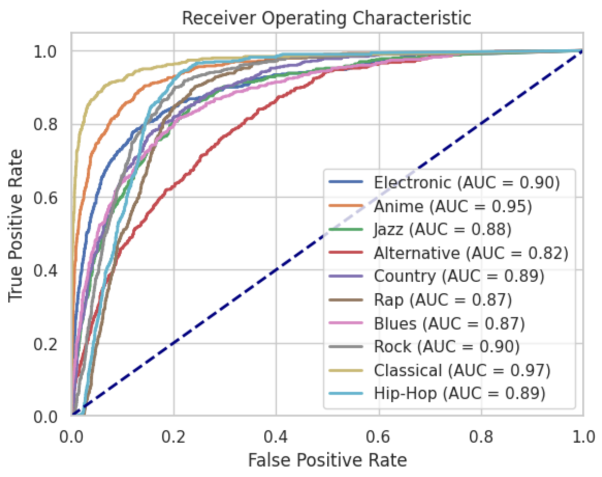


An AUC of 0.95 means that this model performs very well in distinguishing between the different genres. Although the classifier itself would have a big impact on performance, I think my choices in pre-processing and dimensionality reduction contributed to a large portion of the success of the model as well. Firstly, I was careful about data leakage occurring between the training and test sets by ensuring that the test set was untouched and no information in the test set was used during training through the imputation and normalization methods I applied to the dataset, as explained earlier in the report. Thus, it helped build confidence that the AUC score determined by my model was high not as a result of bias. In addition, my decision to utilize 8 components rather than 2 during the dimensionality reduction stage also was an important factor with the performance of my model. If I used a 2 dimensional training set to train the classifier, it probably would have resulted in a poor AUC score as the cluster visualization showed the lack of separation between the different genres of songs.

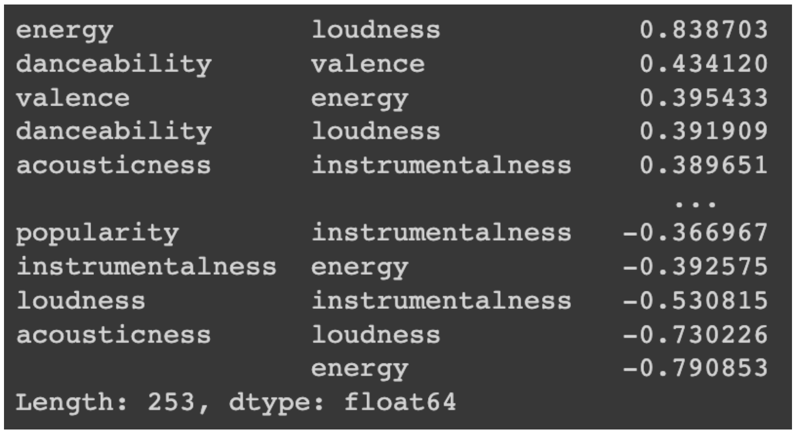
Random Forest Classifier

I also chose to try a Random Forest Classifier as well. It performed just as well as SVM in terms of AUC score, most likely because random forest handles non-linear and high-dimensional data as well. However, I chose to use SVM over random forest in my main report because the individual AUC scores of each genre are less varied in SVM. In the graph shown below, there are some genres where the AUC scores are significantly higher than others but in SVM, the individual AUC scores are closer together.





I also looked at the correlation matrix of the dataset to see how correlated different features of a song are. As I expected, most of the features are not strongly correlated with one another.



```
energy            loudness             0.838703
danceability      valence              0.434120
valence           energy               0.395433
danceability      loudness             0.391909
acousticness      instrumentalness     0.389651
                                         ...
popularity        instrumentalness    -0.366967
instrumentalness  energy              -0.392575
loudness          instrumentalness    -0.530815
acousticness      loudness            -0.730226
                  energy              -0.790853
Length: 253, dtype: float64
```

With 23 different features of a song used to classify its genre, there were 253 correlation coefficients. The screenshot above shows these correlation coefficients in descending order, and out of these 253, only 5 pairs have a correlation coefficient above |0.5|. Hence, this explains why when using 2-component PCA to reduce the data, songs are nearly indistinguishable from another. In other words, using only 2 components does not give enough information to accurately predict the genre of a song because most features of a song are not correlated with one another, making it hard to reduce the dimension of the data without losing information.