The first step in the data cleaning and preprocessing phase is to address the issues present in the raw data obtained from the source. This data contains a lot of missing values and incorrect entries, requiring careful handling.

To begin, I loaded the data and sampled 20 rows to gain a general understanding. I noticed the presence of 'nan' and '?' values, which I replaced with NaN for further cleaning. Instead of simply dropping this data, I aimed to retain as much information as possible.

Since the instance_id, artist_name, and track_name variables are not relevant for the subsequent classification, I set them as the index. In the tempo column, I observed the presence of '?', which I also replaced with NaN. To ensure consistency, I converted all non-NaN values to numeric and filled the NaN values with the average tempo value.

For the columns: popularity, acousticness, danceability, duration_ms, energy, instrumentalness, liveness, loudness, tempo, and valence, all of which are numeric, I checked for outliers and examined their range. It came to my attention that the duration_ms column had outliers with a value of -1.0. I addressed this by replacing it with the average value of the column.

As for the categorical columns, namely key and mode, I obtained their unique values and performed one-hot encoding. Following the encoding, I dropped the original columns to eliminate redundancy.

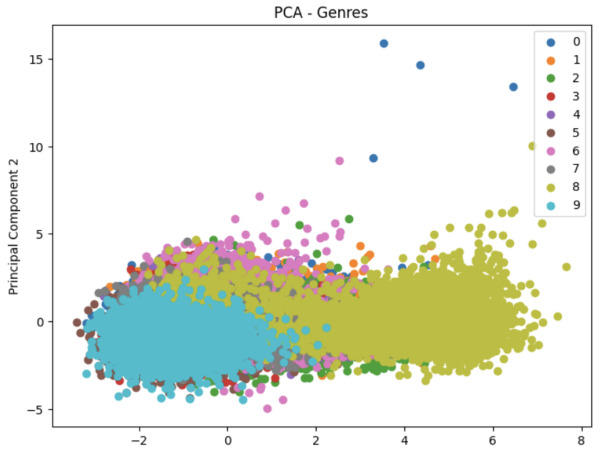
Lastly, I examined the unique labels present in the 'music_genre' column. After completing the cleaning and preprocessing steps, the data now appears as follows:

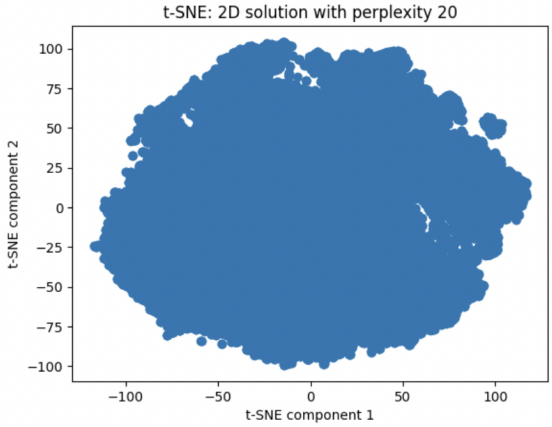
```
Data columns (total 26 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   popularity        50000 non-null  float64
 1   acousticness      50000 non-null  float64
 2   danceability      50000 non-null  float64
 3   duration_ms       50000 non-null  float64
 4   energy            50000 non-null  float64
 5   instrumentalness  50000 non-null  float64
 6   liveness          50000 non-null  float64
 7   loudness          50000 non-null  float64
 8   speechiness       50000 non-null  float64
 9   tempo             50000 non-null  float64
 10  valence           50000 non-null  float64
 11  music_genre       50000 non-null  int64
 12  key_A             50000 non-null  uint8
 13  key_A#            50000 non-null  uint8
 14  key_B             50000 non-null  uint8
 15  key_C             50000 non-null  uint8
 16  key_C#            50000 non-null  uint8
 17  key_D             50000 non-null  uint8
 18  key_D#            50000 non-null  uint8
 19  key_E             50000 non-null  uint8
 20  key_F             50000 non-null  uint8
 21  key_F#            50000 non-null  uint8
 22  key_G             50000 non-null  uint8
 23  key_G#            50000 non-null  uint8
 24  mode_Major        50000 non-null  uint8
 25  mode_Minor        50000 non-null  uint8
dtypes: float64(11), int64(1), uint8(14)
memory usage: 8.7+ MB
```

Then, I moved onto dimension reduction. I decided to use PCA and t-SNE. The reason for my decision is because they are both commonly used dimension reduction methods that provide nice visualization. Before I applied the dimension reduction, I separated out the numeric features and the one-hot-coded feature. Because PCA and t- SNE usually works with standardized data so I have to standardize the numeric features.

```
# numeric features
numeric_features = ['popularity', 'acousticness', 'danceability', 'duration_ms',
                    'energy', 'instrumentalness', 'liveness', 'loudness',
                    'speechiness', 'tempo', 'valence']

# One-hot encoded features(No standarization needed)
one_hot_features = ['key_A', 'key_A#', 'key_B', 'key_C',
                    'key_C#', 'key_D', 'key_D#', 'key_E', 'key_F',
                    'key_F#', 'key_G', 'key_G#', 'mode_Major', 'mode_Minor']

X_numeric = df[numeric_features].values
X_one_hot = df[one_hot_features].values
y = df['music_genre'].values

# Standardize the numeric features
scaler = StandardScaler()
X_numeric_std = scaler.fit_transform(X_numeric)


X_std = np.concatenate((X_numeric_std, X_one_hot), axis=1)
```

The results of PCA and t-SNE dimension reduction techniques are presented below. The PCA analysis provides valuable insights into the clustering patterns of the data. Additionally, I conducted a detailed analysis of the dimensions by examining the correspondence of the PCA features.In particular, the pca_component1 demonstrates relatively high loadings for features such as acousticness, energy, loudness, and instrumentalness. This suggests that songs with higher values along pca_component1 tend to have greater acousticness, lower energy, lower loudness, and higher instrumentalness.On the other hand, the pca_component2 exhibits higher loadings for features like danceability, liveness, and speechiness. This indicates that songs with higher values along pca_component2 tend to have lower danceability, higher liveness, and higher speechiness.





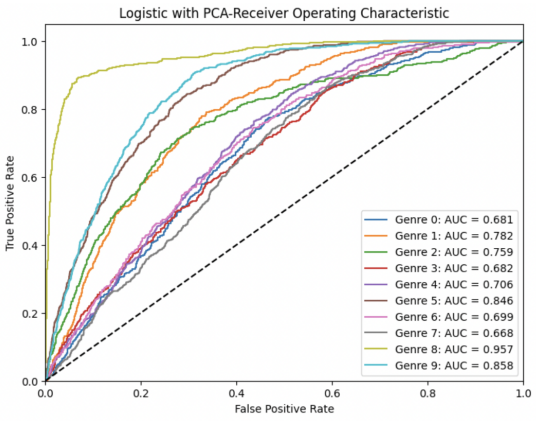
The plot generated using t-SNE did not provide significant insights. I attribute this limitation to the selection of inappropriate hyperparameters, which affected the visualization. However, considering the size of the dataset, the run time of t-SNE became very long when

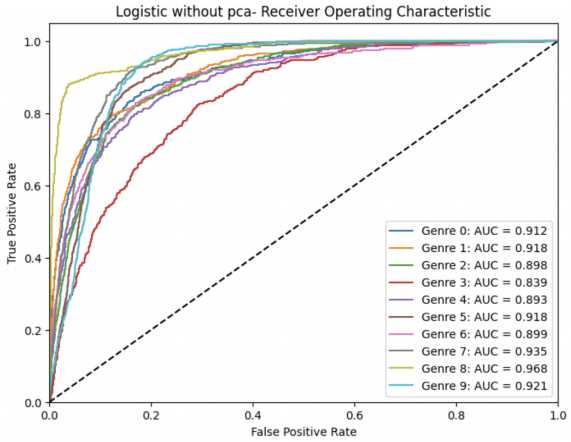
trying different perplexity values. As a result, I decided to rely on PCA for dimension reduction due to its efficiency in handling larger datasets.

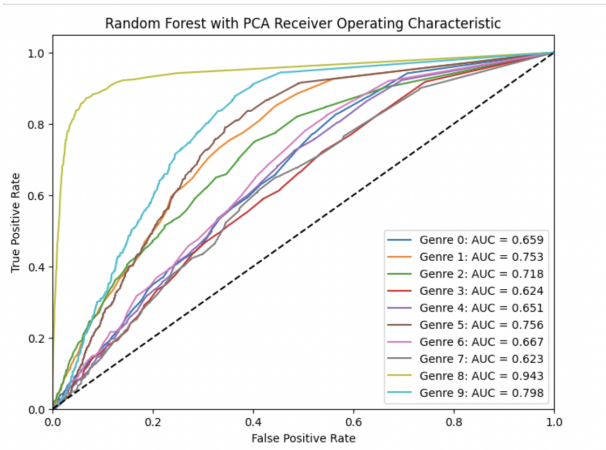
I decided to use Logistic regression and Random forest classification for classifying the data. I used a PCA reduced test set as well as test value from original data.

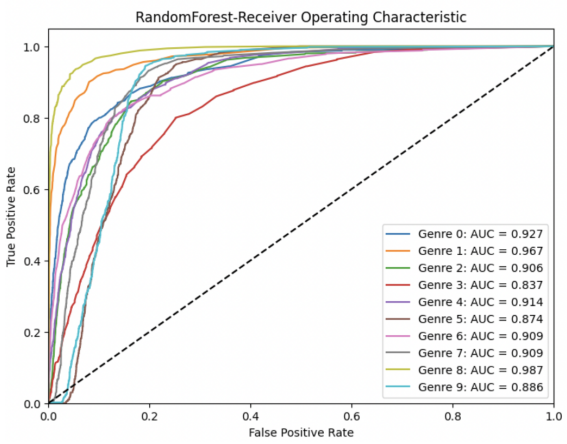
Model	Accuracy	AUC
Logistic Regression after Dimension Reduction	0.2862	0.763873
Logistic Regression without Dimension Reduction	0.5482	0.910113
RFC after Dimension Reduction	0.2480	0.719372
RFC without Dimension Reduction	0.5266	0.902874

I found the logistic regression and the RFC with the original data performed better models using PCA reduced data. This makes sense because our data is already labeled,meaning it contains information about the target variable. By using the original data directly, the models have access to all the features and their relationships, allowing them to better capture the patterns in the data. In contrast, PCA reduces the dimensionality of the data by projecting it onto a 2-dimensional space, in this case, potentially losing some information.









Extra Credit:

I wrote code to calculate the average loudness of each artists' song and top 5 are:

```
Top 5 Artists by Average Loudness:
Coachwhips: 1.949
Lil Texas: -0.40275
kradness: -0.5566666666666668
King Khan and the Shrines: -0.748
SVDDEN DEATH: -0.867
```

YAHHH SVDDEN DEATH

and....let me show you one of my t-SNE runs: