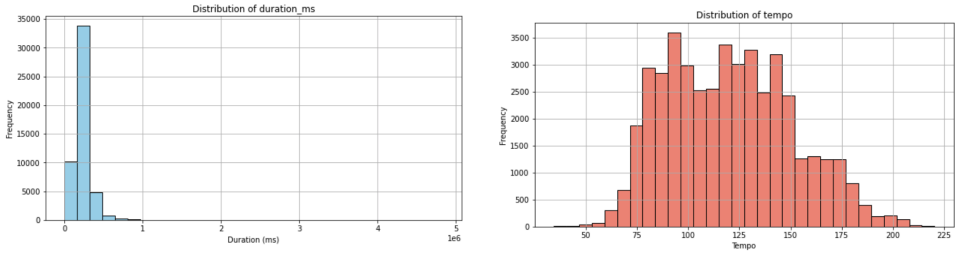
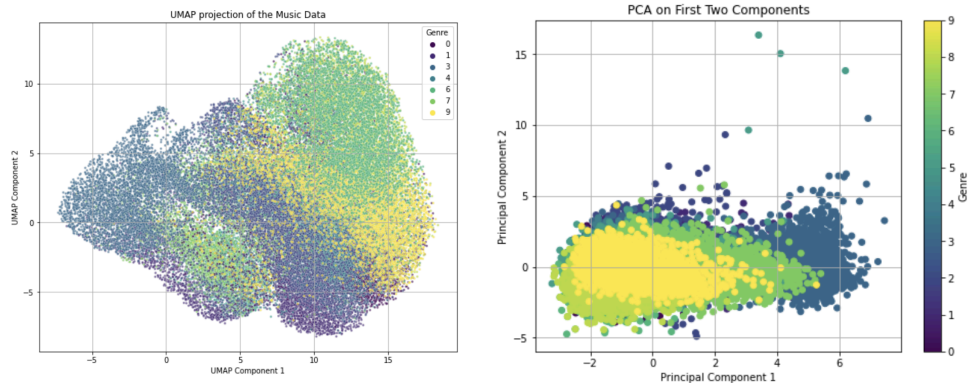
1. Data Cleaning and Imputation and Normalization:

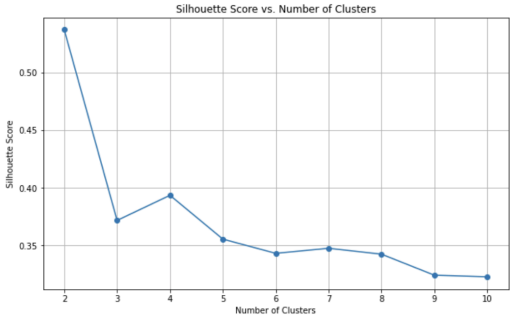


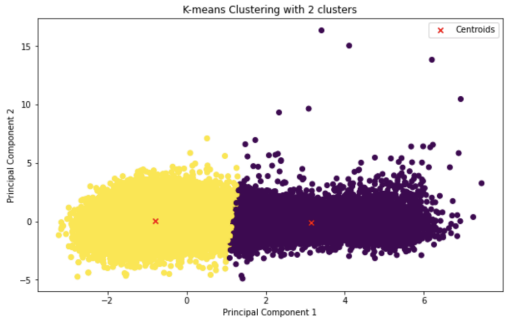
The initial step in preparing the musicData.csv dataset involved several data cleaning and imputation techniques to address missing values and non-numeric data types. First, the five rows with null values were dropped. For the 'tempo' feature, which included question marks as placeholders, these were identified and replaced with the mode of the column after evaluating its distribution. Similarly, any negative values in the 'duration_ms' column were considered errors, set to NaN, and then imputed with the column's mode. Subsequently, irrelevant features such as 'instance_id', 'artist_name', 'track_name', and 'obtained_date' were removed from the dataset as they were not essential for the analysis. The remaining numeric features were standardized using Z-score normalization, except for categorical features like 'mode' and 'key'. These categorical variables were one-hot encoded but not normalized. The target variable 'music_genre' was transformed into categorical labels, facilitating the model training process. Finally, to ensure robust model evaluation, the dataset was split into training and test sets. Specifically, for each music genre, 500 songs were randomly selected to form the test set, and the remaining 4500 songs were used for the training set. This division guaranteed that the test set included 5000 songs, with 500 from each genre, ensuring a balanced representation and preventing data leakage between the sets.

2. Dimension Reduction Using PCA and UMAP: I tested both Principal Component Analysis (PCA) and UMAP (Uniform Manifold Approximation and Projection) to reduce the dimensionality of the dataset. Although both methods aimed to project the data into a 2D space for easier visualization and analysis, the results were disappointing as the projections did not clearly separate the data points as seen in the graphs below. Between the two, PCA was notably faster, which influenced my choice to proceed with it despite the similar performance in dimensionality reduction. Using the Kaiser criterion, which retains components with eigenvalues greater than 1, I identified three significant principal components. Consequently, I decided to use PCA with these three components for the subsequent clustering process. This approach allowed me to reduce noise and focus on the most informative features, enhancing the overall efficiency of my analysis.

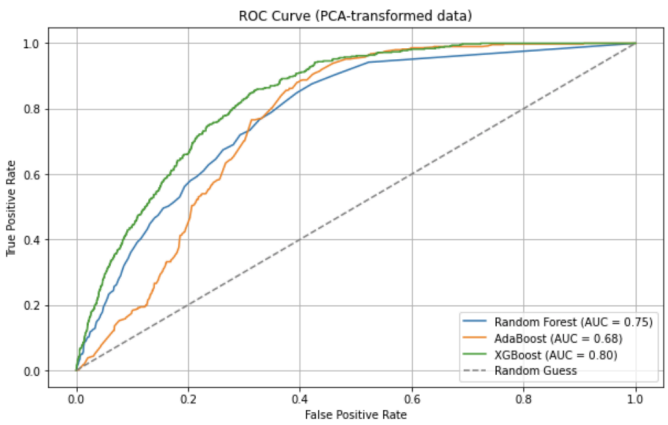


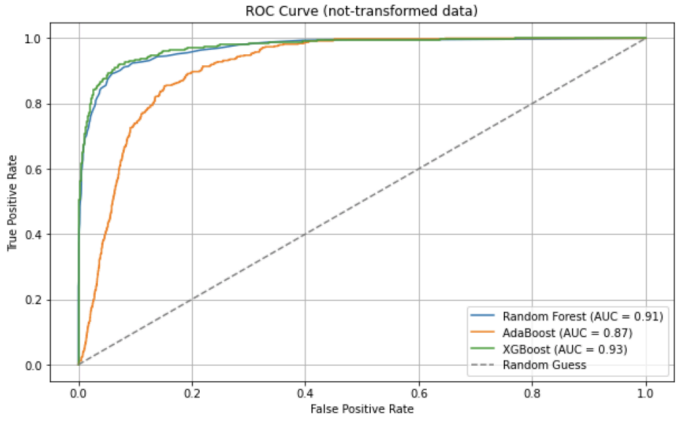
3. Silhouette Scores and Clustering: To validate the effectiveness of the PCA and explore potential groupings within the data, I performed K-means clustering on the PCA-transformed data. To find the optimal number of clusters, I used silhouette scores, which measure how similar an object is to its own cluster compared to other clusters. The silhouette plot indicated that two clusters were optimal. However, this result is not ideal for our purpose, as we know the dataset contains 10 different genres. As shown below, the K-means clustering, visualized using only the first two principal components, confirmed two clusters, which is not accurate for representing the variety of genres. Given this discrepancy and the unsuccessful clustering into the expected ten genres, I will proceed with other modeling approaches without relying on the clustering results from K-means. This decision is based on the clear indication that K-means did not capture the complexity of the genre distinctions in this dataset.





4. Experimenting with Different Models: I evaluated several classification models on both PCA-transformed and non-transformed data to determine the most effective approach for our dataset. Below we can see the AUC of different models with PCA-transformed data and Non-transformed data. Here's a concise comparison of their performances: Random Forest, AdaBoost, and XGBoost all showed significantly better results on non-PCA data as compared to PCA-reduced data, with XGBoost leading in accuracy (0.5766) and AUC (0.9307). Neural Network also performed well on non-PCA data, with an accuracy of 0.5634 and an AUC of 0.9193, closely following XGBoost. Given these results, **XGBoost** stands out as the best choice. It not only provided the highest accuracy but also maintained good performance on PCA-transformed data. Its robust handling of complex datasets and efficient processing make it especially suitable for this task.





5. Best Model and Hyperparameter Optimization: XGBoost emerged as the best-performing model due to its high accuracy and AUC score. To further refine the model, RandomizedSearchCV was utilized for hyperparameter optimization, incorporating cross-validation to prevent data leakage. This method not only searches across a specified grid of parameters but also ensures that each combination is validated in a robust manner through cross-validation. **The final AUC achieved on the test set was 0.933**, reflecting high model performance. The most effective parameters are detailed in the image below, along with the accuracy and AUC of the final model used for classification. This approach of using RandomizedSearchCV with cross-validation was pivotal in optimizing the model to its best potential.

```
Fitting 5 folds for each of 50 candidates, totalling 250 fits
Best Parameters: {'n_estimators': 200, 'max_depth': 3, 'learning_rate': 0.2}
Test Set Metrics:
Accuracy: 0.5916
AUC: 0.93362
```

Conclusion: The success of the XGBoost classification model is primarily due to its robust handling of complex data structures, XGBoost's capability to manage intricate data makes it ideal for nuanced tasks like music genre classification. Cross-validation in the hyperparameter tuning process further reduces overfitting and enhances the model's generalizability to new data.

6. Extra-Credit: For extra credit, I plotted the cosine similarity between the 10 music genres in our dataset. The plot revealed both dark and lighter lines indicating varying degrees of similarity and considerable overlap between different genres. This visualization underscores the complexity of the classification task. The significant similarities between genres contribute to the lower accuracy of our classification models, as it complicates the ability to distinctly categorize each genre. This overlap explains why even the most effective models, such as XGBoost, face challenges in achieving higher accuracy levels. This aspect of genre similarity is crucial for understanding the limitations and challenges inherent in classifying music based on these features. Having domain knowledge is immense for such a classification task.