# Optimizing kNN and Random Forest Models for Music Genre Classification Through Feature Engineering and Dimensionality Reduction

**Introduction to Machine Learning CS 601.475**

**Team Members:**

- Prabuddha Ghosh Dastidar
- Lukas Geer
- Adrian Archer

**December 2024**

# 1. Introduction

As the year ends many of us are bombarded with numerous yearly recaps across social media. Of these recaps, whether photo dumps of vacations or lists of favorite books, the most pervasive is the music recap from companies like Spotify and Apple. In these analyses of listening habits from across the year a consistent staple of the personalized report is that of the "Top Genres" category. While this is a simple matter of summing the genres assigned to songs listened to and sorting by total time there are always questions raised every year as to the accuracy of these listings. Someone who identifies as a devout blues listener may be pinned as a country fanatic or a hip-hop head will be labeled as a soul follower. This kind of genre distinction is left to the label's discretion and can dilute the listening experience knowing it will mess with an individual's end-of-year summary while also preventing fans of a certain genre from trying a new song from a different genre that is fundamentally misrepresented. This was the seed by which we decided to explore a machine-learning approach to genre classification of music, going beyond the arbitrary labeling of humans and trying to algorithmically expose the soul of the song.

# 2. Problem Description

From the perspective of the average person receiving their musical recap we have a disconnect between the heard experience and the arbitrary genre assignments placed on their favorite songs. While an artist may benefit from blurring the lines between genres for targeting a specific Grammy nomination or demographic this is a source of dishonesty which is a disservice to the listener. This is our first reason for pursuing this paper.

Numerous fiery debates have been had over the genre of a piece is spite of being provided a genre classification by the artist of the label. As such, we aim to provide an impartial third-party voice to settle these disagreements with a holistic, machine learning driven analysis of songs to produce a genre classification that does not exclude consideration of any features relevant to a song from vocals to bass to drums. This is our second reason for pursuing this paper.

There is also great merit to pursuing audio analysis through machine-learning from an academic perspective. Audio is a data type with much greater complexity than text and is a current frontier of machine learning. Our genre analysis will be using cutting-edge techniques in t-SNE and UMAPS to enhance fundamental models in classification, these being K-NN and random forests. Creating enhanced versions of cornerstone models will allow for novel comparison and a jumping point for new experimental approaches as we display later in the paper. This is our third reason for pursuing this paper.

In all, we aim to move the world of genre classification and music analysis forward in application and theory via the creation, analysis, and comparison of fundamental models enhanced via feature manipulation and modern frameworks in t-SNE and UMAPS.

# 3. Data and Features

The GTZAN11 dataset is a musical dataset consisting of 10000 3 second audio files of songs of 10 different genres: blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae, and rock. The features are extracted from librosa a multitude of musical attributes like spectral features (which are basically musical fingerprints that tell us information about the different frequencies present in the sound).

The following is a description (taken from the dataset itself) of the features:

- **Spectral Centroid**: This feature measures the "center" of the frequency spectrum, often associated with the brightness of the sound. Higher values correspond to sharper, treble-heavy sounds, while lower values correspond to bass-heavy sounds.
- **Spectral Rolloff**: Spectral rolloff indicates the frequency below which a certain percentage (usually 85%) of the signal's total energy lies. Lower values typically indicate a darker, bass-heavy sound, while higher values suggest a brighter sound.
- **Spectral Flux**: This measures how much the spectrum changes from one frame to the next. High spectral flux indicates rapid changes in the signal, commonly found in percussive or transient sounds.
- **Spectral Bandwidth**: Spectral bandwidth measures the width of the spectrum. A larger bandwidth indicates a richer, more complex sound, while a smaller bandwidth suggests a simpler or more tonal sound.
- **Zero Crossing Rate**: This feature counts the number of times the signal crosses zero. A high zero crossing rate indicates a noisy or percussive sound, while a low rate is typical of smoother, continuous sounds.
- **MFCC (Mel-frequency Cepstral Coefficients)**: MFCCs represent the spectral shape of the signal in a way that mirrors human auditory perception. They capture the timbre of the sound and are useful for music analysis.
- **Chroma Feature**: Chroma features relate to the twelve pitch classes in music (e.g., C, D, E, etc.), capturing harmonic content. They are useful for understanding tonal structures such as key and chords.
- **Root Mean Square Error (RMSE)**: RMSE measures the energy or loudness of the signal. A higher RMSE value indicates a more dynamic or energetic sound, while a lower value indicates a quieter or steadier sound.
- **Harmonic-to-Noise Ratio (HNR)**: HNR compares the harmonic (musical) content to the noise (non-harmonic) content. A higher HNR suggests a more tonal sound, while a lower HNR indicates more noise or less harmonic content.
- **Autocorrelation**: Autocorrelation measures how similar a signal is to a delayed version of itself. It is used to detect periodicity, such as repeating rhythms or sections in the music.

It is important to note that the first twenty MFCCs were used. Additionally, for every attribute in the list above, its mean and variance were separate features.

All data was standardized (centered at zero and divided by the standard deviation) before use (except only centered for PCA). In an exploratory analysis, we found that the variance-based features have a right-skewed distribution across samples. In order to amplify differences in variance across samples (and consequently give these features a normal distribution), all variance features had an IHS (inverse hyperbolic sine) transformation applied to them.

We also found use in splitting the audio files into different audio components and extracting these features for each audio component (refer to section 7).

# 4. Methods

kNN and Random Forest classifiers were applied to the full data, and then the data after its dimensionality was reduced via t-SNE, PCA, or UMAP. In this project, all dimensionality reduction techniques were applied with the final embedding being 3 dimensional.

This analysis was done on the original GTZAN data (after preprocessing) and the data after features were split into drums, bass, vocals, and other (see section 7).

## 4.1. PCA

PCA or Principal Component Analysis is a dimensionality reduction technique used in machine learning to simplify datasets while maintaining as much variability as possible. It works by identifying the principal components, which are directions in the data that capture the maximum variance. PCA transforms the original data into a new coordinate system where these principal components serve as the axes. This is especially useful for reducing noise, speeding up computation, and visualizing high-dimensional data. However, PCA assumes linear relationships and may not perform well with highly non-linear data.

## 4.2. kNN

k-Nearest Neighbors (kNN) is a simple and intuitive algorithm used for classification and regression tasks. It works by finding the k closest data points (neighbors) to a query point based on a chosen distance metric (e.g., Euclidean distance). The algorithm predicts the output by taking a majority vote for classification or averaging the values for regression among the neighbors. kNN is non-parametric, meaning it makes no assumptions about the data's underlying distribution. However, it can be computationally expensive as it requires storing all training data and calculating distances for each query.

## 4.3. Random forests

Random Forests is a versatile ensemble learning algorithm primarily used for classification and regression. It works by building multiple decision trees during training and combining their outputs, typically through majority voting for classification or averaging for regression. Each tree is trained on a random subset of the data and features, introducing diversity and reducing the risk of overfitting. This approach makes Random Forests robust, accurate, and effective at handling non-linear relationships. However, it can be resource-intensive, and its interpretability is lower compared to simpler models.

### 4.4. t-SNE

t-SNE11, an enhancement of Stochastic Neighbor Embedding (SNE), helps us uncover patterns in high-dimensional data by focusing on pairwise similarities. We start by measuring the Euclidean distances between data points. For each point, we place a Gaussian distribution centered on it, adjusting its width to reflect the local density of points. In regions with higher density, the Gaussian is narrower. We then evaluate how well each point aligns with others by projecting these distances onto the Gaussian, where the height of the curve represents the similarity.

To simplify computations, we transform these similarities into probabilities, normalizing them so they sum to one. Instead of analyzing all point pairs, we use a perplexity parameter to decide how many neighbors to focus on for each point. This parameter sets a target density, influencing the scale of the Gaussian for each point. The conditional probability that a point $x_j$ is selected relative to $x_i$ is calculated as:

$$p_{j|i} = \frac{\exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(-\frac{\|x_i - x_k\|^2}{2\sigma_i^2}\right)}$$

Here, $\|x_i - x_j\|$ is the distance between $x_i$ and $x_j$, and $\sigma_i$ relates to the perplexity for $x_i$. After computing these probabilities, we construct a similarity matrix for the dataset. To make the matrix symmetric, we adjust the probabilities as follows:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$$

Next, we map the high-dimensional points into a lower-dimensional space, initializing the positions randomly. In this space, we measure similarities using a Student's t-distribution, chosen for its heavier tails, which help distribute points more effectively. The probabilities in the embedded space are calculated as:

$$q_{j|i} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_{k \neq i} (1 + \|\mathbf{y}_i - \mathbf{y}_k\|^2)^{-1}}$$

Here, $\|\mathbf{y}_i - \mathbf{y}_j\|$ represents the distance between points $\mathbf{y}_i$ and $\mathbf{y}_j$ in the lower-dimensional space. Our aim is to align the high-dimensional similarity distribution $p$ with the low-dimensional one $q$. Initially, the two distributions don't align, so we iteratively refine the positions in the lower-dimensional space.

To achieve this, we minimize the Kullback-Leibler (KL) divergence, which quantifies the difference between $p$ and $q$:

$$D_{\mathrm{KL}}(P_i \parallel Q_i) = \sum_{x \in X} p_i(x) \log\left(\frac{p_i(x)}{q_i(x)}\right)$$

By summing the divergences across all points, we define a cost function:

$$C = \sum_{i=1}^{N} D_{\mathrm{KL}}(P_i \parallel Q_i)$$

We use gradient descent to minimize $C$. The gradient of $C$ with respect to $y_i$ is:

$$\frac{\partial C}{\partial y_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij})(y_i - y_j)(1 + \|y_i - y_j\|^2)^{-1}$$

Here, $p_{ij}$ and $q_{ij}$ are the pairwise similarities in the original and embedded spaces, respectively. By adjusting the positions iteratively, we reduce $C$, refining the low-dimensional representation.

This process enables us to capture local relationships in the data, clustering similar points together in the low-dimensional space. However, t-SNE prioritizes local structure over global relationships, meaning the relative distances between clusters may not always represent meaningful relationships.

### 4.5. UMAP

Uniform Manifold Approximation and Projection (UMAP)11 operates in a manner similar to t-SNE. To begin, we construct a graph based on k-Nearest Neighbors (kNN). Using a similarity metric $d$, such as the Euclidean distance, we define $\rho_i$ as:

$$\rho_i = \min\{d(x_i, x_{i_j}) \mid 1 \leq j \leq k, d(x_i, x_{i_j}) > 0\}$$

Here, $\rho_i$ represents the smallest non-zero distance between $x_i$ and its nearest neighbor. Next, we determine $\sigma_i$ such that the following condition holds:

$$\sum_{j=1}^{k} \exp\left(-\frac{\max(0, d(x_i, x_{i_j}) - \rho_i)}{\sigma_i}\right) = \log_2(k)$$

By ensuring $\rho_i$ connects each point $x_i$ to at least one neighbor with a weight of 1, we maintain the assumption that our data is uniformly sampled on $\mathcal{M}$. This assumption is critical for UMAP's ability to handle high-dimensional data efficiently, which is supported by principles of fuzzy topology. From these parameters, we compute the weights as:

$$w_{ij} = \exp\left(-\frac{\max(0, d(x_i, x_{i_j}) - \rho_i)}{\sigma_i}\right)$$

These weights quantify the probability of a connection between two points. If we represent the weighted similarity matrix as $A$, we construct a symmetrized matrix $B$ using:

$$B = A + A^T - A \circ A^T$$

where $\circ$ represents the element-wise product of matrices.

In the optimization step, we solve a non-convex problem using an approach similar to simulated annealing. Although random initialization is possible, we typically use the symmetric Laplacian of the graph (spectral embedding) for better stability and speed. We optimize the layout of points, $\{y_i\}_{i=1}^n$, to minimize the following cross-entropy cost function:

$$C_{UMAP} = \sum_{i \neq j} v_{ij} \log\left(\frac{v_{ij}}{w_{ij}}\right) + (1 - v_{ij}) \log\left(\frac{1 - v_{ij}}{1 - w_{ij}}\right)$$

To achieve this, we apply forces during optimization: attractive forces along edges and repulsive forces on vertices. The attractive force is given by:

$$\frac{-2ab\|y_i - y_j\|^{2(b-1)}}{1 + \|y_i - y_j\|^2} w_{ij}(y_i - y_j)$$

where $a$ and $b$ are tunable parameters. The repulsive force is expressed as:

$$\frac{2b}{(\epsilon + \|y_i - y_j\|^2)(1 + a\|y_i - y_j\|^{2b})}(1 - w_{ij})(y_i - y_j)$$

Here, $\epsilon$ is a small constant (commonly 0.01) to prevent division by zero. During each iteration, we adjust points based on both attractive and repulsive forces. Over time, the optimized graph $\{y_i\}_{i=1}^n$ aligns with the structure of the original data $\{x_i\}_{i=1}^n$, preserving its topology and enabling us to visualize the data effectively in lower dimensions.

# 5. Initial Results

Table 1: Accuracy Comparison

| **Model Type** | **Accuracy** |
|---|---|
| Random Forest | 87 |
| kNN | 45 |
| Random Forest with PCA | 51 |
| kNN with PCA | 46 |
| Random Forest with UMAP | 85 |
| kNN with UMAP | 85 |
| Random Forest with t-SNE | 93 |
| kNN with t-SNE | 92 |

As we can see, reducing the dimensionality of the GTZAN dataset using t-SNE and then applying Random Forests to classify genres based on the features being t-SNE components resulted in the best accuracy in these initial results. This accuracy was 93 percent.

It is important to note that t-SNE used a perplexity parameter of 10, which is relatively low in the world of machine learning. This emphasizes that the local structure of the data (compared to global) is very important for classification by genre. This may also be why t-SNE was the most effective dimensionality reduction technique, because it focuses deeply on preserving local structure of the data compared to UMAP and PCA which focus on global structure.

Additionally, it is important to note that reducing the dimensionality of the dataset using t-SNE enhanced the performance of the two classifiers. This is possibly because the dimensionality reduction worked to reduce overfitting and prevent the model from 'memorizing' the training data.

Refer to the figures showing the confusion matrix for the best model (Figure 2) along with the associated 3D embedding (Figure 1).

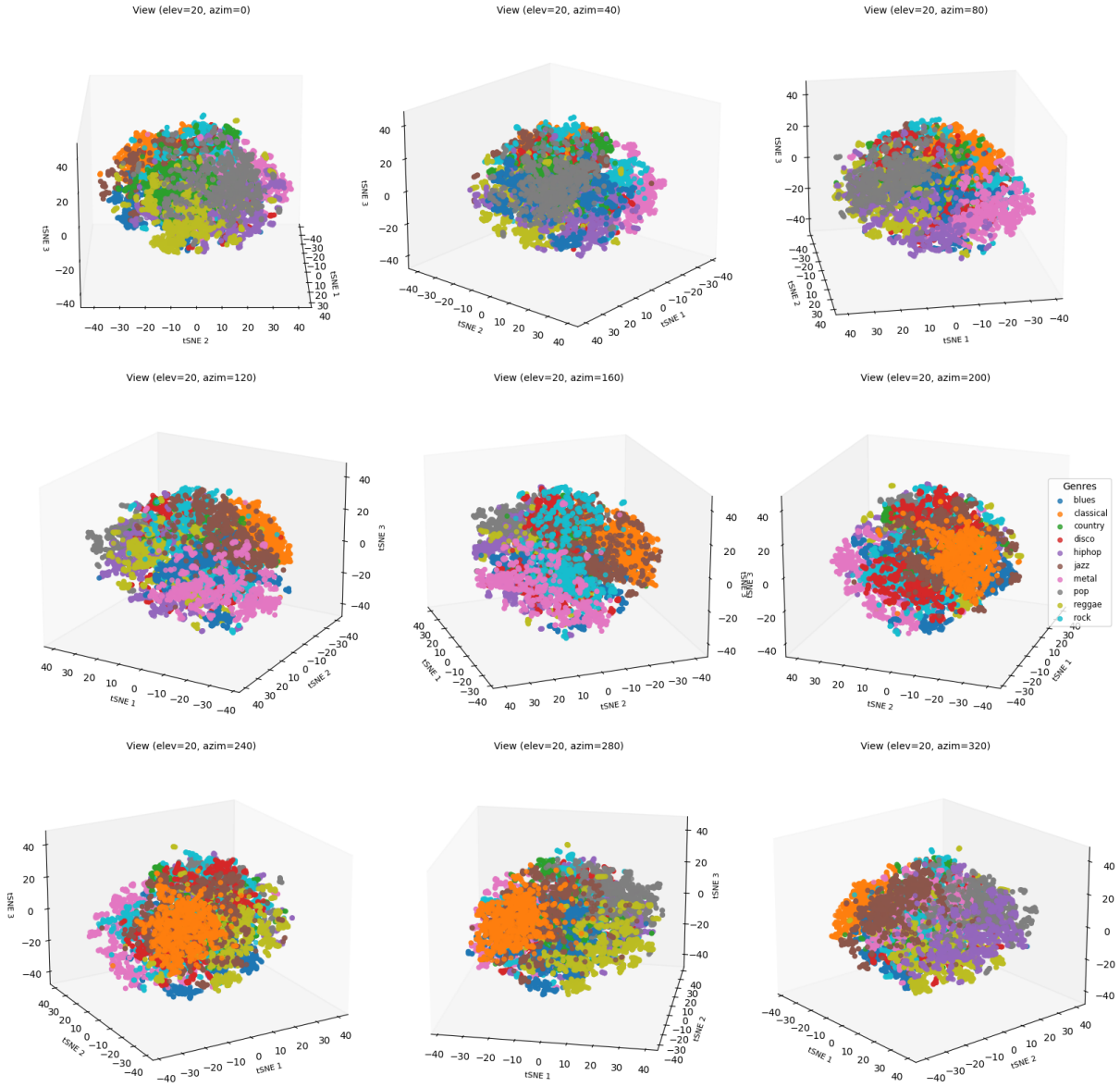


Figure 1: 3D t-SNE Embedding of GTZAN Data

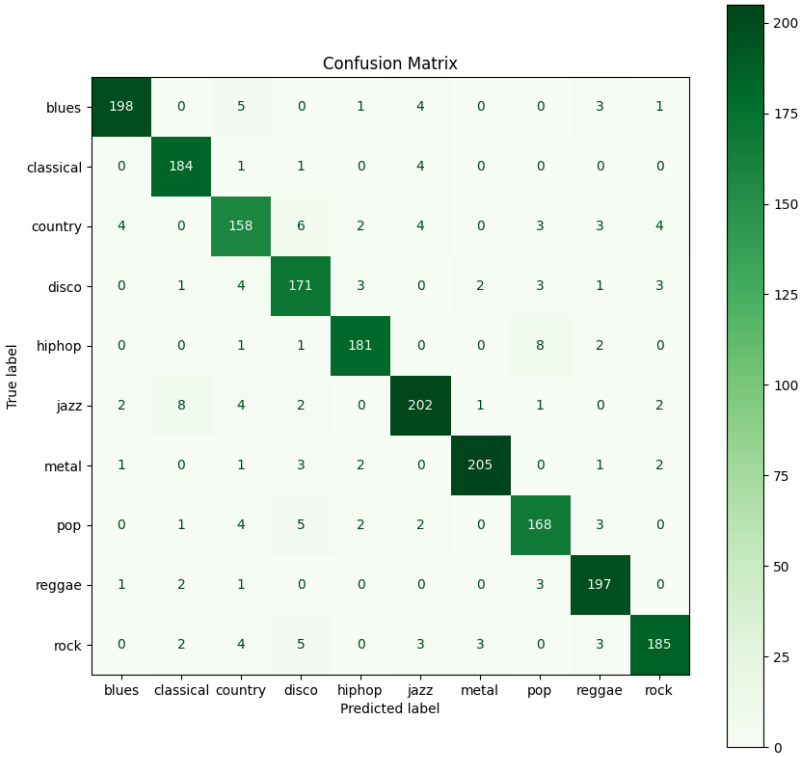


Figure 2: Confusion Matrix for Random Forest with t-SNE on GTZAN Data

# 6. The Locality Conjecture

Given that our model's performed at a higher level with t-SNE dimensionality reduction over UMAP and that our kNN model performed best with a k value of 3, we reasoned that our models had the highest accuracy when prioritizing the local structure of the data. That is, in a elementary explanation, the closer the similarity(or distance) between two data points, the more likely they are to be in the same genre. To optimize this similarity emphasis, we hypothesized that our misclassified data points may be due to the close similarities between some of the components of genres. For instance (this is not proven to be correct), but classical and jazz may tend to have very similarity audio features corresponding to certain components. We can logically infer this from our confusion matrix from our initial models, which tend to confuse genres which are observationally similar. Furthermore, we proposed that these similar components between the genres may overpower other components in our original audio files, so our models which prioritize global structuring (UMAP and kNN with higher k) ignore other components important to differentiating the genres.

# 7. Experimental Model and Final Results

**Basis:**
Building on our locality conjecture, we decided to pursue an experimental avenue which arises naturally from our conjecture of pre-processing our data into their components (splitting the data). We reasoned that if we split the audio files into their musical components, our models could highlight these characterizing differentiations as the separated data would decrease the distraction of any one component and allow them to focus on the features of the components which separate the genres.

**Implementation:**
To split our data, we opted to use another framework called demucs11, which splits the data into four component files: drums, bass, vocals, other. Once these files were split, we then ran the librosa11 feature extraction software to extract the spectral features of the files. Since the components of the same track ended up on different rows of the csv file, we had to do some additional modifications to move the features from the different components corresponding to the same audio file onto the same row. Finally, we ran new extracted data points through each of our models in the exact same combinations as before and received promising results.

**Results:**

Table 2: Accuracy Comparison

| | Random Forest | kNN | Random Forest with t-SNE | kNN with t-SNE |
|---|---|---|---|---|
| Original Data | 87 | 45 | 93 | 92 |
| Split Data | 91 | 64 | 97 | 95 |

For comparison, we tried to find the models with the best recorded accuracy on the GTZAN dataset. Both ANN(artificial neural network)11 and CNN had high accuracies of 92.44% and 93.65%11 respectively, which are very decent numbers. This shows the significancy of our new model that outperforms these high accuracy models by almost 4%.

Refer to the figures showing the confusion matrix for the best model (Figure 4) along with the associated 3D embedding (Figure 3).

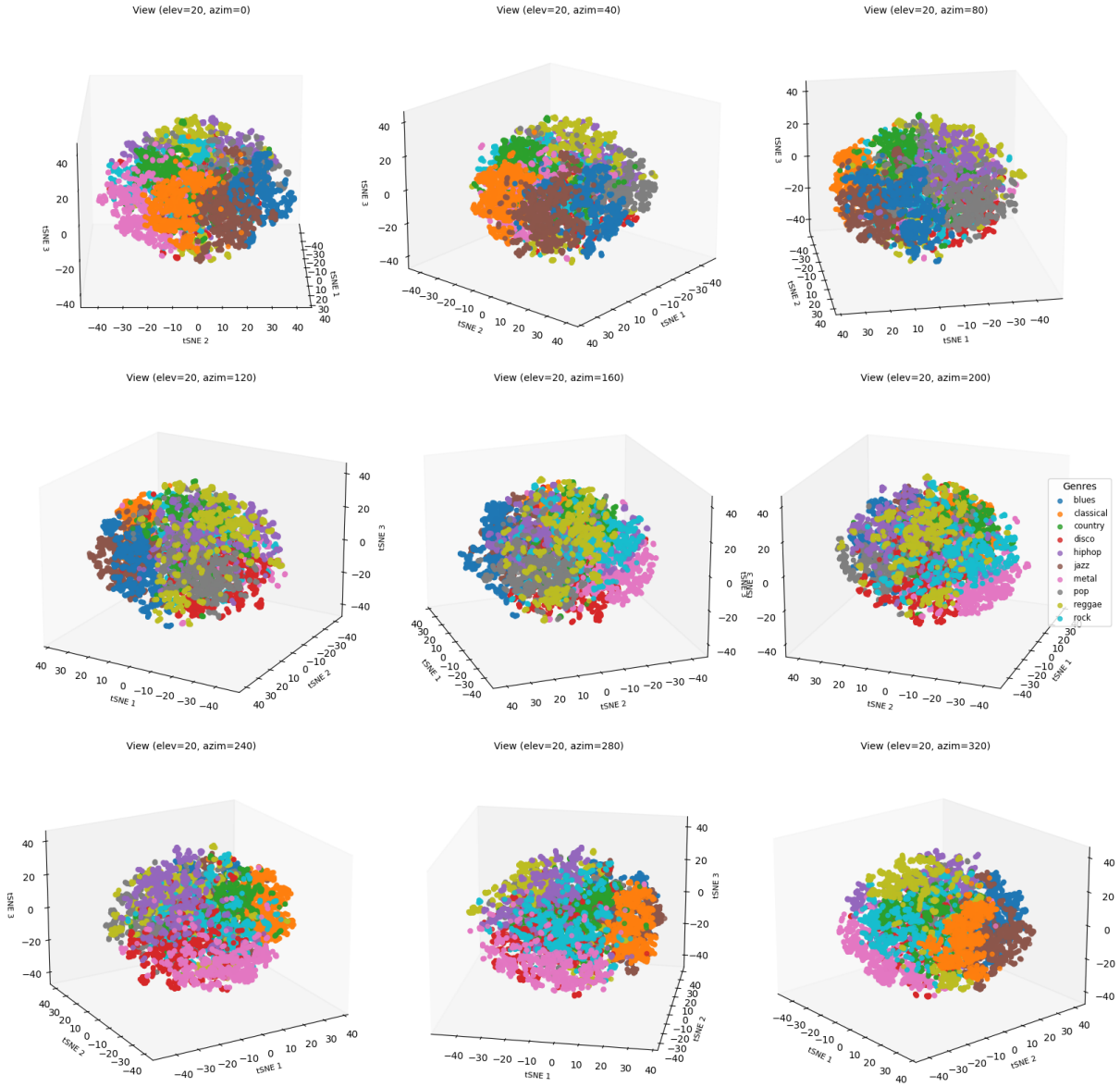


Figure 3: 3D t-SNE Embedding of GTZAN Data Where Features Were Split By Audio Component

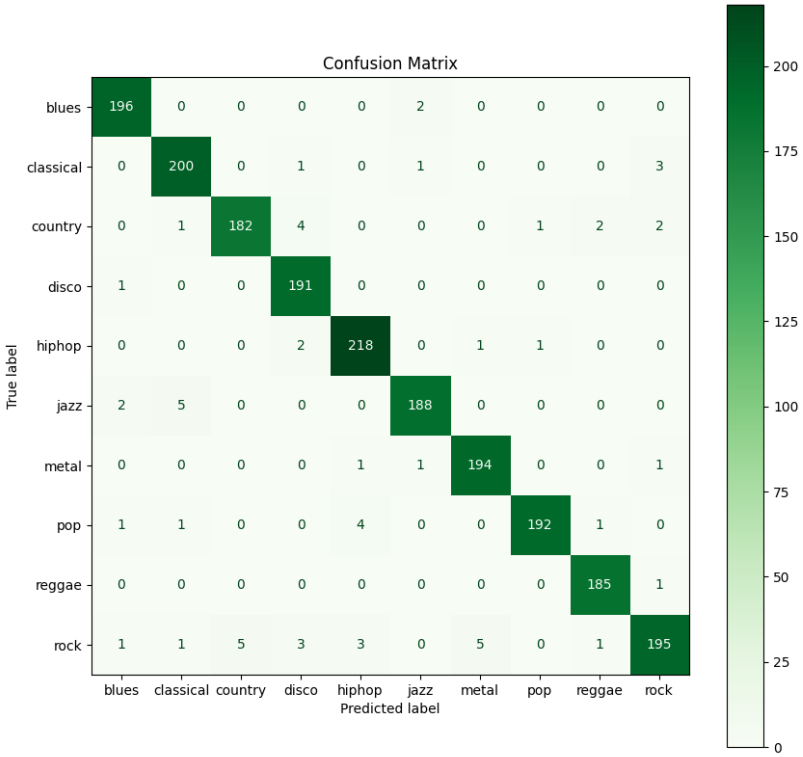


Figure 4: Confusion Matrix for Random Forest with t-SNE on GTZAN Data Where Features Were Split By Audio Component

# 8. Difficulties encountered

**Runtime:**
We don't possess the optimal resources to run these computationally complex models like librosa feature extraction and demucs in a short amount of time. Thus, it took us an extended amount of time to run these models and proved to taxing especially when their runtimes quit a few times. The free subscription tier of google colab notebook doesn't give you the ability to run cells while your computer is closed, so we eventually opted to purchase the pro version so this was possible.

**Dimensionality reduction methods:**
Initially, in the beginning stages of our model selection and data pre-processing plan development, we decided to use PCA. We were expecting this to at least improve the accuracy of our models by a small amount. However, our accuracies decreased, which was quite discouraging to see. Instead of giving up on the use of dimensionality reduction in our model entirely, we thought it would be a good idea to try some unconventional methods for the dimensionality reduction of audio data. After some research and exploration, we came upon the algorithms of UMAP and t-SNE and thought it would be good

to compare with each other because of their goals of retaining global and local structure, respectively.

**Model development:**
Once we determined that our project's new course of action would be to try to develop our models in a novel way, we realized it would be a difficult task to come up with such a development. Looking at the current models for genre classification and comparing them with random forests and kNN, there wasn't a clear path to improvement without the use of neural networks, which we decided not to use because of our limited resources. Thus, it took us a while to come up with an effective approach to improve our existing models. Eventually, we did conjure up a novel way of bettering our model through the splitting of the audio files as a form of pre-processing.

**Processing of audio files:**
This was not mentioned in the previous sections, but the GTZAN dataset only comes in the form of 30 second audio files. So, in order to compare the accuracy of the models on the original 3-sec feature csv file versus the split 3-sec features, we had to separate the 30 second audio files into 3-second sections while retaining their proper ordering relating to track number so that we could connect them back to their corresponding track numbers when creating the pandas dataframe containing the data points. This presented a very tedious process which took a long time to figure out.

# 9. Future Work

Our paper clearly delineates three paths by which this research can be expanded. These paths are the optimization of feature extraction, the expansion of the locality conjecture, and the use of new models and frameworks to expand the record of comparisons.

First, for the optimization of feature extraction we suggest a more sophisticated separation of songs into their audio components. We obtained incredible results with a separation of principle audio into their drum, vocal, bass, and "other" components but we can break these components further into fundamental pieces like melody, timbre, dynamics, and texture allowing for a more complex analysis that can properly separate the few stubborn errors such as comparisons between heavy metal and rock. It would also be interesting to explore features that are specific to instrumental or vocal-based music to expand distinctions within genres such as breaking classical down into specific eras such as contemporary, baroque, or romantic.

Second, for the expansion of the locality conjecture, we have evidence and a conclusion that neatly ties together the power of local preferences in analyzing musical features. This opens the door to exploring proof-based reasoning as to why this conjecture prevails and gives a hint about what models should be tried first in expanding this work; those that have a greater affinity for localized interpretations of their input data.

Third, we could take the split data and use PCA or similar variance maximizing techniques to embed this data in lower dimensional space, and then attempt to cluster by genre and cluster by audio component. This will give us insight into whether the audio component or the musical genre is the main source of variation in the musical data and

could open doors to more complex feature analysis.

Lastly, for the use of new models and frameworks in expanding the scope of our comparisons we have shown 11 unique approaches to our original classification problem and leave room for dozens of other approaches. We touched on the highest accuracy we have observed from a CNN but there is clearly more room for improvement on the neural network front. There are also exciting ramifications of a novel audio analysis approach with t-SNE that opens the door to applying this framework to other models beyond kNN and random forests.

# 10. Conclusion

In this paper, we have worked through numerous combinations of models and frameworks to provide a highly accurate method for classifying genres for music addressing the three initial motives for pursuing this area. We were also able to produce the locality conjecture, which we have strong reasoning and evidence to believe in. There is also an extremely promising set of research areas that can be directly expanded upon from the paper. In all, we believe this is a significant finding in musical analysis inspired by a modern societal experience we share every year and a debate that has existed for centuries; what genre is that song?

# 11. Citations


1. Artificial Neural Networks (ANN): (2024). A neural network model for music genre classification with GTZAN dataset. arXiv preprint, 2410.14990. Retrieved from https://arxiv.org/html/2410.14990v1

2. Convolutional Neural Networks (CNN): Wang, P., Zhang, L., Li, J., Yang, X. (2021). A novel middle-level feature learning method for music genre classification using convolutional neural networks. Electronics, 10(18), Article 2206. https://doi.org/10.3390/electronics10182206

3. Demucs (audio splitter): StemRollerApp. (n.d.). StemRoller [Computer software]. GitHub. Retrieved December 18, 2024, from https://github.com/stemrollerapp/stemroller

4. GTZAN dataset: Olteanu, A. (n.d.). GTZAN Dataset - Music Genre Classification [Data set]. Kaggle. Retrieved December 18, 2024, from https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classificationc

5. Librosa: McFee, B., Matt McVicar, Daniel Faronbi, Iran Roman, Matan Gover, Stefan Balke, Scott Seyfarth, Ayoub Malek, Colin Raffel, Vincent Lostanlen, Benjamin van Niekirk, Dana Lee, Frank Cwitkowitz, Frank Zalkow, Oriol Nieto, Dan Ellis, Jack Mason, Kyungyun Lee, Bea Steers, ... Waldir Pimenta. (2024). librosa/librosa: 0.10.2.post1 (0.10.2.post1). Zenodo. https://doi.org/10.5281/zenodo.11192913

6. UMAP: McInnes et al., (2018). UMAP: Uniform Manifold Approximation and Projection. Journal of Open Source Software, 3(29), 861, https://doi.org/10.21105/joss.00861

7. t-SNE: Van der Maaten, L., Hinton, G. (2008). Visualizing data using t-SNE. Journal of Machine Learning Research, 9(Nov), 2579–2605. http://www.jmlr.org/papers/v9/vandermaaten08a.html