

[Tyler]

Hey Everyone,

T: My name is [Tyler]

G: and my name is [George]

Thanks for checking out our presentation for our Intro to ML - Final Project Presentation.

Our group had a keen interest in revolving the project around the topic of music and audio, and were able to find an interesting dataset to allow for some Exploratory Data Analysis using some methods and topics touched upon in lectures across the semester.

[George]

The focal point for our project is to provide EDA for a Musical dataset that consists of various music genres. This dataset includes .wav files, which are the highest-resolution of digital audio formats, and a list of extracted features of each audio track in the dataset.

Next, we'll provide an overview for what this presentation will touch upon.

Table of Content

- Objective & Dataset
- Features of Audio Data
- Initial Work
- Model Selection
- Optimization Methods
- Classification Results
- Further Considerations
- References

[Tyler]

The outline for our presentation can be seen in the following list provided in this slide.

To provide some background to those who may not be as familiar with audio, or as a refresher to others, we'll start the presentation by explaining the dataset we acquired and cover the various types of features that were extracted from the raw audio samples, and also present our group's objectives for EDA.

We'll then dive into our initial qualitative analysis to better understand the dataset. We started off the initial work of the project to gain an intuitive understanding of how different genres of music can be presented to differentiate one musical category from another. Some of this included calculating the most crucial audio features that help make these distinctions, of which we'll touch upon.

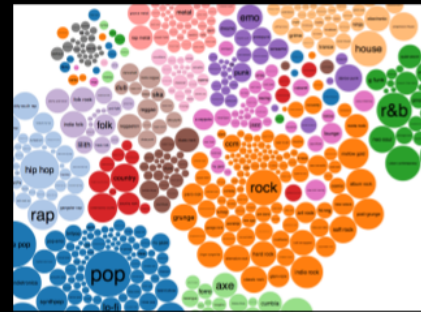
[George]

Then we'll discuss the types of classification models we decided to experiment with and describe the methods of optimizing not only the splitting of the Test/Training datasets, but how we optimized more complex models that require finer tuning of their parameters.

The results from our classification methods will then be presented in a table and graphical format to help examine our analysis of all the models we've chosen, and then we'll conclude with some further considerations our group investigated into for additional areas of exploratory data analysis

EDA Objective & GTZAN Dataset

- **GTZAN Dataset:**
 - 1000 audio files from 10 different genres, each 30s long.
 - CSV file containing the different features of each audio file.
- **EDA Goals:**
 - Determine which classification model yields the best performance.
 - Determine which features are most useful for classification.
 - Relate these features to the different properties of musical genres.



Clusters of Music Genres [9]

[Tyler]

The dataset we decided to perform EDA methods with is called the GTZAN dataset, and it contains 1000 audio files in .wav format from 10 different genres of music, each 30 secs long. The list of genres include:

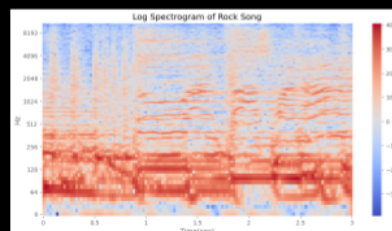
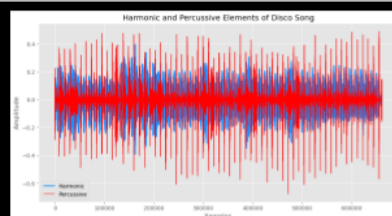
- Blues, Rock, Hip-Hop, Pop, etc

There were 100 songs for each genre category. This dataset includes songs from artists such as Beyonce, David Bowie, Iron Maiden, Bob Marley, etc... So, these are well-known artists and chart-topping songs from across the previous 60 years give or take. Included for a particular song from a genre, there are about 57 features that were extracted for each audio sample both from time-domain and frequency-domain which characteristically describe the waveform and spectrum of the audio.

Our goals for EDA were to determine which classification algorithms could be trained to provide the best prediction accuracy for all 10 genres of the GTZAN dataset. For optimizing the classification model, we also hoped to discover which features served as the most useful parameters and how they help differentiate one genre from another

Features of Audio

- 57 Features in Dataset
 - Mono .wav audio files
 - Sample Rate: 22,050 Hz
- Extracted using Librosa Module
 - Time-Domain
 - Tempo
 - Harmonic/Percussive Elements
 - Frequency-Domain
 - Log Spectrum
 - Bandwidth
 - Spectral Centroid
 - Chroma (Notes A - G)
- Implement methods of Dimensionality Reduction and Feature Selection to gain knowledge and aid classifiers



[Tyler]

The audio was formatted to Mono .wav files which contain no compression that are found in sound formats such as .mp3s with a sample rate of 22,050 Hz

Features were extracted from each audio .wav file using the Librosa Library in Python, that has numerous functions that compute features of audio primarily from both Time-Domain and Frequency-Domain or Spectral information

Features such as the Tempo of a song, as well as the Harmonic and Percussive elements were extracted from the Time-Domain. An example of this can be seen from Disco Song in the top figure.

Likewise a song's Frequency-Domain information provided a majority of the feature set, for instance a Log Spectrogram of the audio, which shows the intensity of the audio's frequency components versus time in terms of a logarithmic scale. Can think of this as FFT at each time instant

From spectral information, features like the Spectral Bandwidth, the range of frequencies a song is composed of, and the Spectral Centroid, which is the mean value taken from the bandwidth

A number music theory-based features were extracted that could describe which musical notes, like the Chroma - chromatical notes A through G on a piano, for which melodies or chords could fall into 1 of 12 possible bins.

Starting with all these features, it made sense that our group first implement methods of dimensionality reduction to provide us some visualizations for the dataset and then feature reduction to gain a better understanding of the extracted features in the context of aiding our classification model, and which ones matter the most for optimizing models

Dimensionality Reduction

- **Principal Component Analysis**
 - Reduced 57 Features down to 2 Principal Components
 - Tradeoff in dimension reduction to provide intuitive visualization for separation of genres
- **K-Means Clustering**
 - Genre labeling is abstract
 - Unsupervised approach to gain understanding of how similar some genres are
 - Reduced 10 genres into 5 clusters after applying PCA



[Tyler]

Starting with the 57-dimension feature space of the dataset, we used Principal Component Analysis to reduce the feature-space down to the 2 Principal Components for each song. From here, we could see from the top figure that genres like Metal (pink) and Classical (orange) were each mapped to certain areas of this 2D graph which shows that the songs in these genres are composed around certain harmonic and dynamical characteristics. While, other genres like Rock and Blues have higher variance in their Principal Component distributions and are scattered throughout the entire plot. As avid music listeners, this lines up with how abstract genre labels can be for certain types of music from one another. Genres like Country and Rock music branched off from the melodic compositions of Blues music, and similarly Hip-hop and Pop music drew influence from the beat and rhythmic dynamics of Disco music.

So, there tends to be a gray area in music where genres influenced other genres, and certain categories of music are considered evolutionary from a former category. So, we used an Unsupervised learning approach where instead of using the provided genre labels, we used the K-Means Clustering algorithm to calculate just how different every song in the dataset is from one another. From these methods, it was estimated that instead of 10 genre labels, all the songs in our dataset could actually be sorted into 5 different clusters. This clustering of the Principal Components displays that the time and spectral characteristics of the songs in our dataset are actually more similar than they are different according to a genre label that either the artist identifies as, or we as listeners identify a certain artist to be categorized into. We see this as the audio data speaking for itself rather than an artist misclassifying their own song.

So, we then turned to examination of the features provided in the dataset

Initial Feature Selection

- **Basic Feature selection:**
 - ran Naive Bayes using a single feature to classify between all 10 genres
 - able to obtain an initial ranking by iterating through features
 - Average accuracy: 0.201
- **Issues:**
 - Only evaluates 1 feature at a time
 - Assumes independence, (generally not true)

Top 5 Features (single-feature classification)	
Feature:	Accuracy:
spectral_bandwidth_mean	0.283
chroma_stft_mean	0.280
rolloff_mean	0.269
perceptra_var	0.267
spectral_centroid_mean	0.245

[George]

Although PCA is powerful, it is a little abstract. We wanted to get an idea of how individual features actually contributed to the accuracy results. For our preliminary analysis, we used a basic feature selection method. We ran Naive Bayes with a single feature, and classified between all 10 genres. We then iterated through all 57 features of the dataset, and ordered these features by accuracy to obtain a basic ranking of feature performance. The top 5 features are shown in the slide. The top-performing features were the spectral bandwidth mean and the chroma STFT mean, which both performed about 8% better than the average.

While this method does allow us to get a quick and easy ranking of the features, its biggest limitation comes from its simplicity. Since we are only evaluating one feature at a time, we cannot take into account the effect of different combinations of features, which is a crucial factor when classifying with more than one feature.

Additionally, there are limitations inherent in the model used to make these classifications, as Naive Bayes assumes independence between the features. This is generally not true, and so a more robust classifier should be used in order to account for this issue.

Sequential Feature Selection

- Expands on our initial, basic Feature selection:
 - Start with 0 Features
 - adds the feature that gives the best performance
 - Stop when we hit a specified # of Features.
- Allows us to get a sense for the most important Features, as well as performance vs # of Features
- Drawbacks:
 - Not as efficient as PCA
 - Computationally intensive



[George]

With these initial results, we attempted to improve upon our initial feature selection method. We did this through the use of the sequential feature selection method, specifically the Sequential Forward Floating Selection algorithm.

This algorithm starts with 0 features, then adds the feature that leads to the best performance, measured in this case by the accuracy of the classifier, similar to our previous method. However, it then repeats this process until we hit the number of features specified (in this case, all 57 features in our dataset). It should be noted that we are using the floating version of this algorithm, which also checks during this process if removing a feature leads to an increase in performance.

(While this might seem counterintuitive (generally more data = better performance), some features might actually be misleading, like in the previously mentioned case. This additionally allows us to check a greater number of subsets, at the cost of speed.)

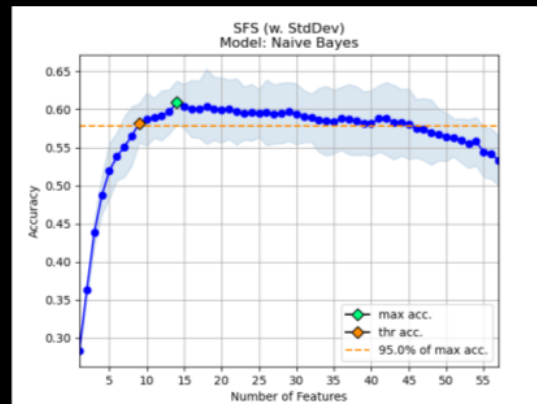
The use of SFS allows us to understand how accuracy changes as we increase the number of features, and lets us get an intuition for which combination of features leads to the best performance.

However, SFS does have its drawbacks. Firstly, it is not as good at feature reduction as PCA. A feature may give great performance between one pair of genres, and an actively negative performance between another. PCA avoids this problem by taking the best components of each feature, but SFS is unable to do this. Additionally, SFS, especially the floating algorithm, is very time consuming, as we are running the classifier over and over again. For a fast classifier and a small number of features, this is not too much of an issue. However, this method would most likely be unfeasible if using a more intensive classifier or a large number of features.

Sequential Feature Selection

Example using Naive Bayes:

- More Features - not necessarily better!
 - Accuracy using all features: 0.533
 - Max accuracy: 0.609, at 14 features
 - ~95% of max. accuracy: 0.582, at 9 features

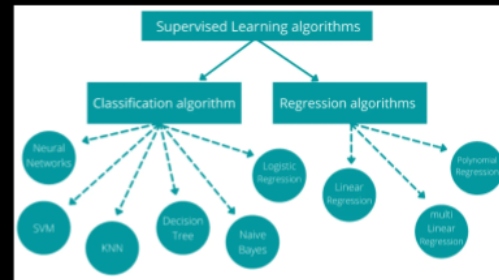


[George]

The slide shows an example of the output of SFS, ran using the Naive Bayes classifier and scored using the accuracy of the classifier. (Results were obtained using stratified K-fold cross-validation, with 5 folds.) As we can see, initially accuracy increases as we increase the number of features. However, after a certain point (14 features), we actually see a decrease in accuracy. For this classifier, features added past this point are not actually informative and overfit the model. Using only 14 features gives us an accuracy of 60.9%. This is nearly an 8% increase over the accuracy using all features (53.3%). This point of maximum accuracy is marked on the plot. As an additional measure to prevent overfitting, we also marked the point where the classifier reaches above 95% of the maximum accuracy. This occurs at 9 features, with an accuracy of 58.2%, which is still above the accuracy calculated when using all features.

Model Selection

- Known Labels = Supervised Learning!
- Key Aspects for Model Selection:
 - Performance
 - Complexity
 - Data Size/Dimension
 - Implementation
- From PCA and Feature Selection:
 - Classification is Linear Problem
 - Models with High Bias/Low Variance
 - Easy to implement, but also could require fine tuning
- Classification Models Chosen:
 - Logistic Regression
 - Linear Support Vector Classification
 - K-Nearest Neighbors



[Tyler]

Since we started with a dataset that contains its own known genre labels, the best applied classification models would be Supervised Learning algorithms. When determining which ones to use we had to keep in mind some important factors for model selection. Our highest priority was the performance of the model, so it must be able to provide a high accuracy score for classification. Since we have a dataset that included 57 features, with some modifications on feature selection, the data size and dimension was also considered. Then lastly, the models should be easy to implement so we can conduct our classification experiments.

From our findings with PCA and Feature selection, we made the assumption that the dataset would be best suited for a linear model. We were more motivated to pick models with a higher bias and lower variance, though also wanted to incorporate models that could require some fine tuning so we could utilize optimizational methods.

The main models we chose to highlight for our EDA project were Logistic Regression, a Linear SVM model, and a K-Nearest Neighbor classifier. Other models were considered and will be mentioned towards the end of the presentation. So, for now we will present our methods and results of classification and optimization of these models.

Optimization Methods

Feature Selection/Reduction:

- Previously mentioned
- Optimal Methods for Training/Testing
 - Stratifying training/test samples
 - 10% of test/training for each genre
 - K-Fold Cross Validation
 - Train models with K folds
 - Compute mean accuracy performance
 - Optimal Value N for KNN Model
 - Compute Error Rate for varying values of N - Neighbors
 - Use value N that minimizes KNN model error → N = 6



[Tyler]

As George previously mentioned, methods of deriving the most important features from the dataset were investigated. We planned to show how the classification models performed using all 57-features and then specify the performance of the model when using the optimal number of features for each.

Optimal methods for splitting the data into Testing and Training sets were also conducted. Since we only had 100 songs for each genre, it was important that we stratify the testing and training datasets. This reassured that the testing and training sets weren't randomly selecting more from one genre than another, and guaranteed that each genre made up 10% for both testing and training sets no matter how we chose train/test split ratio (ended up using a 80% train and 20% test ratio).

Another method we used for splitting up the data for testing and training was K-Fold Cross-Validation. We used the value of $K = 5$ for cross-validation when training each classifier, which gave us a better estimate of the model's performance. Rather than training and testing with the same samples, all of our models were trained and tested using 5 folds from the dataset and then the average accuracy scores were calculated along with their standard deviation to give a more robust prediction score for each model.

Lastly, when using the KNN classifier in Python, the default value for the number of neighbors to use is the value of 5. Instead of using this default value, we iteratively trained the KNN classifier with a varying number of neighbor values between 1 and 40 and calculated the error rate for each outcome. From this procedure, it was discovered that the value of 6 for the number of neighbors was the optimal value that minimized the error rate for the KNN classifier, and was then deployed for our classification experiments.

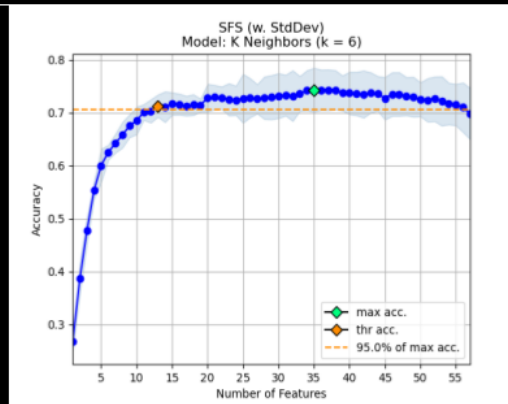
Classification Results

Model	Baseline accuracy (all 57 Features)	Max accuracy (all features)	Threshold accuracy
Naïve Bayes	0.533	0.609 (14 Features)	0.582 (9 Features)
SVM (Linear)	0.687	0.716 (42 Features)	0.681 (26 Features)
Logistic Regression	0.661	0.685 (45 Features)	0.651 (21 Features)
K Nearest Neighbors (N = 6)	0.698	0.743 (35 Features)	0.713, (13 Features)

[George]

The results from comparing different classifiers are above. As we can see, K nearest Neighbors is by far the best-performing classifier, reaching a maximum accuracy of 74.3%. Additionally, it requires less features to achieve this performance than its competition, using only 35 features compared to 42 for SVM and 45 for Logistic Regression. Even when using a reduced number of features (only 13), it still outperforms Naive Bayes and Logistic Regression, and performs comparably to SVM (which is using over 3x the amount of features). It should also be noted that SFS with K-Nearest Neighbors generally took less time than SVM and Logistic Regression, an important note considering how intensive SFS can be.

Classification Results



Sequential Feature Selection Results for K Nearest Neighbors

[George]

The plot of features vs accuracy for the best performing classifier (K Nearest Neighbors, $k=6$) is above. The 95% threshold that we implemented seems to do a good job of avoiding the plateau that occurs after around 15 features.

Feature Results

- For each classifier, we checked the features that SFS selected to give max performance.
- Several features were selected for all 4 classifiers:

Most consistent features:

Chroma_stft_mean
Chroma_stft_var
Mfcc10_var
Mfcc4_mean
Mfcc5_var
Mfcc9_mean
rms_mean
rms_var

[George]

A list of features that contributed to the maximum performance for all 4 classifiers is given in the slide. These are the features that SFS choose for all 4 classifiers. (It's interesting to note that spectral bandwidth mean, our previous choice for a top-performing feature, does not appear here. This is due to the fact that it doesn't actually appear in the SFS-selected features for the K nearest neighbors classifier.) It's also interesting to see the type of features that consistently appear in all 4 classifiers.

- The chroma STFT (short time Fourier Transform) is used to extract pitch information from a song, which can be used to identify things like a song's melody, key, and chord progression. These aspects can vary heavily from genre to genre, so its presence here is unsurprising.
- Several Mel-frequency cepstrum coefficients also show up here. These coefficients have uses in speech processing, and can also be used as an indication of timbre (the quality of a specific sound). Timbre is essentially what allows us to distinguish one instrument from another. Since the timbre of the instruments used is a pretty good indicator of genre, these coefficients do seem to play an important role. (One other thing to note is that our dataset contained 20 MFCCs, but only the lower ones made it to this list. This corresponds with our sources, which mention that MFCCs above 13 are usually discarded as being unhelpful.)
- The RMS value is a measure of the amplitude of a sound's waveform, which corresponds to the volume of the song. Given how loudness and dynamics play different roles in different types of music, it makes sense that the rms mean and variance values would be an important feature.

EDA Considerations



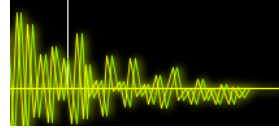
Satisfactory Models

- Decision Trees
- Random Forest

Feature Selection/Reduction

- Lasso Regression: accuracy range: ~50%-60%

Further Improvements

- Neural Networks
 - Traditionally trained with 2D data
 - Methods of using Spectrograms of Audio previously done
 - Increasing Dataset
 - Manually add more .wav files to dataset
 - Splicing audio into smaller clips
- 

[Tyler]

To end the presentation, we wanted to highlight the classifier models and other EDA methods that were considered. Some of the weaker models we used for classification were decision trees and random forest models. These models did not provide noteworthy prediction scores, and generally perform better with smaller feature-sets. Methods of visualizing a decision tree for our dataset were considered, but ultimately were abandoned to focus on the main classifier models we touched upon due to their superior accuracy scores.

Other than Sequential Feature Selection, another method of Feature Reduction we attempted was Lasso Regression. This method uses a technique known as “shrinkage” where the features of the dataset are used and coefficients representing their weight towards classification are shrunk towards zero. After applying Lasso Regression to your feature set, any feature coefficients that are non-zero are kept and the rest are discarded. After performing Lasso Regression and training our classifiers, the accuracy scores were in the range of around 50-60%. The classifiers trained using these features performed consistently worse than the ones trained with the SFS features. So we pivoted to use the method of Sequential Feature Selection instead as it was a better fit for our project objectives.

As far as considering another classifier model, Neural Networks was a model we saw as too ambitious for the time we had left to experiment. Neural Networks traditionally use image information, we considered methods of feeding spectrograms for each song into a Neural Network model, but in the little time we had as a group, we saw this as too demanding. Though, if given more time that would have been the next classifier we considered experimenting with.

Lastly, given that we only had 100 songs for each genre, manually adding 30 sec audio clips of other songs would have been something our group experimented with. Preferably, we would have chosen more songs from the same artists that were categorized in the dataset. Also, a controversial method of splicing our songs into smaller clips in an attempt to enlarge our sample set was considered. But, depending on how repetitive a given song was, this could be viewed as a data leak and used what we were given.

References

1. G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," in *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293-302, July 2002, doi: 10.1109/TSA.2002.800560.
2. L. K. Puppala, S. S. R. Muvva, S. R. Chinige and P. S. Rajendran, "A Novel Music Genre Classification Using Convolutional Neural Network," *2021 6th International Conference on Communication and Electronics Systems (ICCES)*, 2021, pp. 1246-1249, doi: 10.1109/ICCES51350.2021.9489022.
3. Changsheng Xu, N. C. Maddage, Xi Shao, Fang Cao and Qi Tian, "Musical genre classification using support vector machines," 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03)., 2003, pp. V-429, doi: 10.1109/ICASSP.2003.1199998.
4. Venturott, P. H. G. (2021, March 3). *Predicting Music Genres Using Waveform Features*. Medium. Retrieved December 7, 2022, from <https://towardsdatascience.com/predicting-music-genres-using-waveform-features-5080e788eb64>
5. Doshi, K. (2021, May 21). *Audio Deep Learning Made Simple: Sound Classification, Step-By-Step*. Medium. Retrieved December 7, 2022, from <https://towardsdatascience.com/audio-deep-learning-made-simple-sound-classification-step-by-step-cebc936bbe5>
6. Gandhi, S. K. (2021, July 9). *Finding Out Optimum Neighbours (N) Number in the KNN Classification using Python*. Medium. Retrieved December 7, 2022, from <https://medium.com/analytics-vidhya/finding-out-optimum-neighbours-n-number-in-the-knn-classification-using-python-9bdcfeff58e>
7. Pramoditha, R. (2020, December 20). *K-Fold Cross-Validation Explained in Plain English*. Medium. Retrieved December 7, 2022, from <https://towardsdatascience.com/k-fold-cross-validation-explained-in-plain-english-659e33c0be0>
8. Doshi, K. (2021, May 21). *Audio Deep Learning Made Simple: Sound Classification, Step-By-Step*. Medium. Retrieved December 7, 2022, from <https://towardsdatascience.com/audio-deep-learning-made-simple-sound-classification-step-by-step-cebc936bbe5>
9. *Music Popcorn – A Visualization of the Music Genre Space*. Music Machinery. (2013, September 23). Retrieved December 12, 2022, from <https://musicmachinery.com/2013/09/22/5025/>
10. S. Raschka, "SequentialFeatureSelector: The popular forward and backward feature selection approaches (including floating variants)," - mlxtend. http://rasbt.github.io/mlxtend/user_guide/feature_selection/SequentialFeatureSelector/#sequentialfeatureselector-the-popular-forward-and-backward-feature-selection-approaches-including-floating-variants.
11. J. Lyons, "Mel Frequency Cepstral Coefficient (MFCC) tutorial," Practical Cryptography. [Online]. Available: <http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/>. [Accessed: 13-Dec-2022].

[Tyler]

Here are the listed references we used for research and code implementations used throughout our project.



[George and Tyler]

This concludes our Final Project Presentation

Thanks for checking it out everyone!