Team 3: Music for Running

Nathan Luskey, Reagan Matthews, David Wen, & Dylan Reese

Github Repository: https://github.com/nilnate/CS4641_Project

Background

Music is an integral part of many individuals' workout routine. It is shown to improve focus and energy for many. Many Spotify playlists geared towards working out exist for this reason, focusing on songs that are proven to boost mood and energy. For runners, music can be selected based on a song's beats per minute (BPM) to fit their desired pace. Existing playlists are not tailored to individuals' music tastes or specific BPM ranges.

Individuals who seek to become strong runners often begin with an approach that alternates periods of running and walking, called the Run Walk Method (Luff). This is a series of workouts, each consisting of a few minutes of running followed by a period of walking, repeating this cycle for the duration of the workout. This guarantees a slow, but attainable increase in the individual's endurance.

Multiple studies have concluded that the ideal tempo of music for achieving maximum performance in a workout is dependent on the type of exercise being performed. These studies have concluded that an optimal music choice can increase work capacity, as well as workout durations significantly (Markell). Currently, multiple approaches exist using convolutional neural networks (CNNs) with spectrograms and other deep learning algorithms to classify songs into genres or artists. These approaches break the songs down to their amplitudes and frequencies, and then analyze the corresponding time series (Nasrullah &Zhao) (Schreiber & Muller). Additionally, there are multiple applications that exist that attempt to categorize songs for different workouts including: Spring, FITRadio, PaceDJ, etc (Barnwell).

Motivation

Our goal is to generate a playlist tailored to the user's taste with BPMs that correspond to the Run Walk Method's workout routine. Existing applications allow users to select their workout type/tempo, and then provide playlists or songs for end users. Our project will be set apart from these by attempting to sync music tempos with workout intensity, rather than the type of exercise being performed.

Concept

We are proposing a supervised and unsupervised model for training and validation that we will evaluate in parallel.The first machine learning method we are proposing is through supervised learning using decision trees. The labels are available through a dataset from tagtraum(Schreiber) which uses "last.fm, top-MAGD, and

beaTunes” to generate genres. These genres can be analyzed to see if they provide sufficient range of tempo within the genre for a running workout. If the genre categories are good, then we will train and evaluate the decision tree through the methods taught during lecture; when a user wants to generate a running playlist they will be able to take a quiz to trace their way through the decision tree to a leaf node containing the workout playlist.

We use an unsupervised approach as well to see if there are other interesting attributes to the data; the unsupervised learning will be done through PCA and clustering algorithms. The user would be able to submit a playlist which will be evaluated for which cluster it most resembles and the nearest cluster will be used to generate the running playlist.

Our potential contributions after finishing this project are relating song characteristics to genre or category. Both unsupervised and supervised approaches improve the understanding of how basic song characteristics listed in the million song dataset connect to a broader category.

Data

To obtain our data, we tried multiple approaches before finding one that worked well. Initially, we tried using AWS, Amazon Web Services,(“Million Song Sample Dataset.”) to obtain the Million Songs Dataset (this is the recommended method on the datasets website). This gave us a 10,000 song sample dataset, but with many missing attributes. Since we needed a full dataset to train on, this did not work. Some other attempts were made using the Spotipy API, but we did not pursue these further.

After some research, we discovered that a full dataset existed on a data sharing and collaboration platform called Dolthub or Dolt. Dolt is described as the “Git for data”, providing similar version control functionality to Github. The data on Dolthub is stored in relational databases, and can be accessed via the Dolt command line tool. Like Github, these databases are tracked, and states are saved on every commit. Dolt is still in development, with the goal of eventually supporting “100% of the Git command line, and 100% of MySQL commands”. With its current functionality, Dolt is an easy way to gain access to large amounts of data.

The data repository we used on Dolt is called million-songs(“Liquidata / Million-Songs.”). This repo is a full version of the Million Song Dataset containing all attributes, with more than 100,000 rows to train on. A total of 46 attributes exist, most of which are numerical values that can be analyzed.

We used Doltpy, a Python API for running Dolt commands, to implement the MillionSongsAPI. This API interfaces with the million-songs database, allowing for fast data retrieval. The full data retrieval process works as follows: first, the program checks if the user has already cloned the million-songs repository. If not, it is cloned in the

current working directory. Once the directory is stored locally, the user can query the dataset by row and for a specific number of rows. This is accomplished through MySQL commands. Finally, all attributes are parsed to Python data types for easy use. Below is a diagram of the data retrieval process:

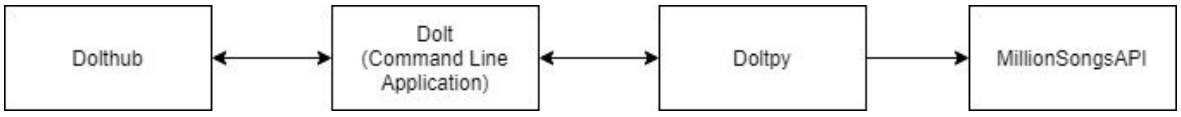

Figure 1. Overview of retrieving data from Dolthub into a local database.

The data retrieval process for the song raw data led to a local database of song data. The genre information was stored in a different file containing unique song identifiers and the genre labels from tagtraum (Schreiber). The file was turned into a hashset with the song as the key and the genre as the value to allow for efficient searching. The songs were then searched to see if they had a genre labeled, and all songs that had a genre labeled were saved to a new compiled csv. We checked for a sufficient range of BPM in different genres and were satisfied with the results shown in figure 2.

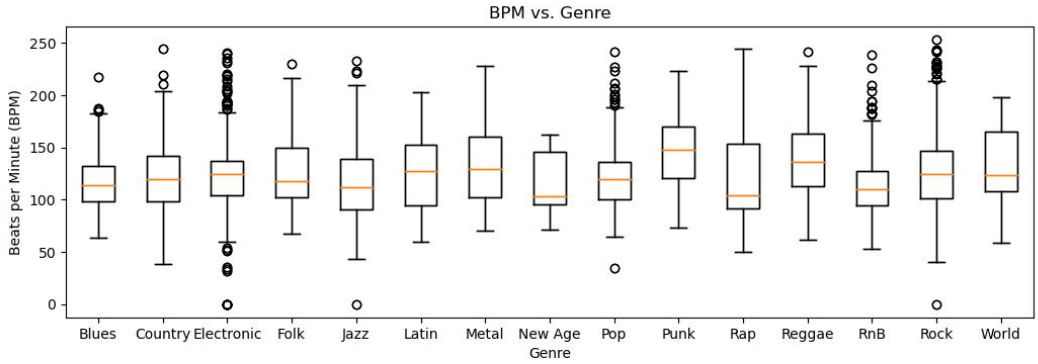

Figure 2. Boxplot of beats per minute (BPM) across the different genres of music.

Unfortunately, we did not realize that unbalanced genres in our dataset could lead to skewed results until after constructing our model. As shown on the pie chart in figure 3, the dominating genre was Rock, with a smaller number of a variety of other genres present.

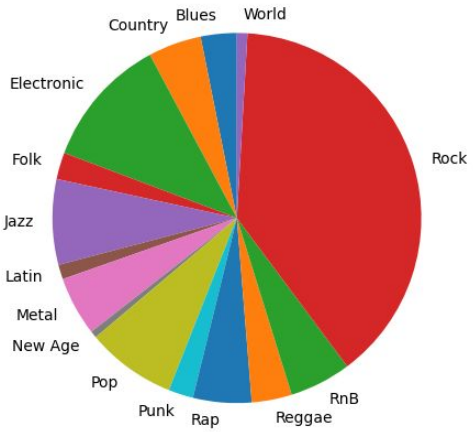

Figure 3. Pie chart of relative representation of genres in the data set.

Motivation for Separate Approaches

To solve the problem of music genre classification, there are many approaches that exist. While some of these approaches may yield better results than others, we decided to focus our attention on one supervised, and one unsupervised learning algorithm. For our supervised approach, we implemented a decision tree with pruning. For the unsupervised one, we performed Kmeans clustering. Each of these approaches analyze the song's attributes, to attempt to further categorize music for audiences with different tastes. Taking these two distinct perspectives on this problem will allow for more insight to be drawn. Additionally, these approaches complement each other well, as the decision tree may provide information on which attributes contribute to the resulting classifications.

Unsupervised Approach

The unsupervised approach uses K-Means clustering(“Sklearn.cluster.KMeans.”) and PCA dimensionality reduction (“Sklearn.decomposition.PCA.”) in scikit. The goal was to group the training data songs into discrete clusters, from which we can pull songs to create our resultant playlist. PCA was used to transform our feature set to speed up the clustering process.

Two feature sets were trained on for this approach. The first set used contained all the numerical features (bars_confidence, bars_start, danceability, end_of_fade_in, etc.), numbering 26 in total. The feature set was reduced down to 10 in total before being clustered. The resultant loss of the trained models for different cluster counts is shown below.

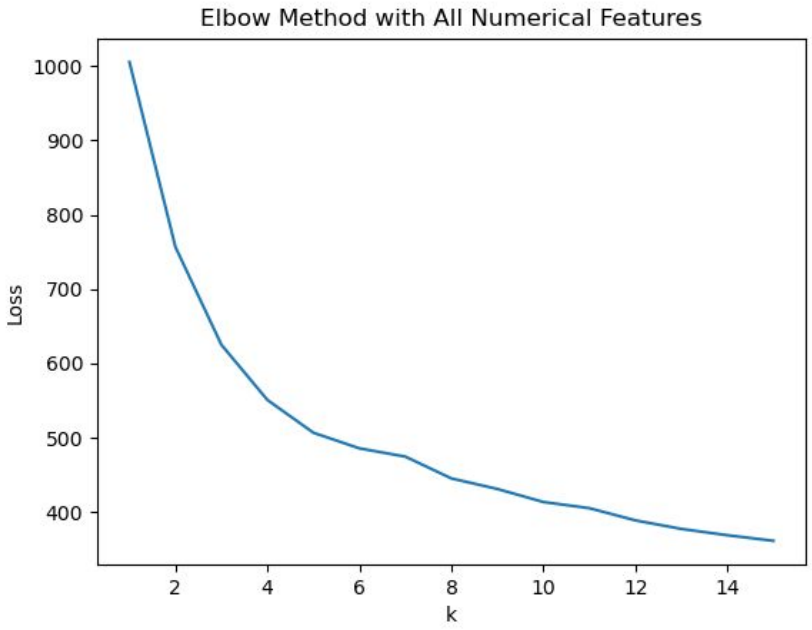

Figure 4. Graph of loss with respect to number of clusters for all features.

The second set used contained only select, important numerical features. This limited set is used in the supervised approach, and contains year, tempo, duration, loudness, mode, and key. For the clustering, the set was reduced down to 3 features. The resultant loss of the trained models is shown below.

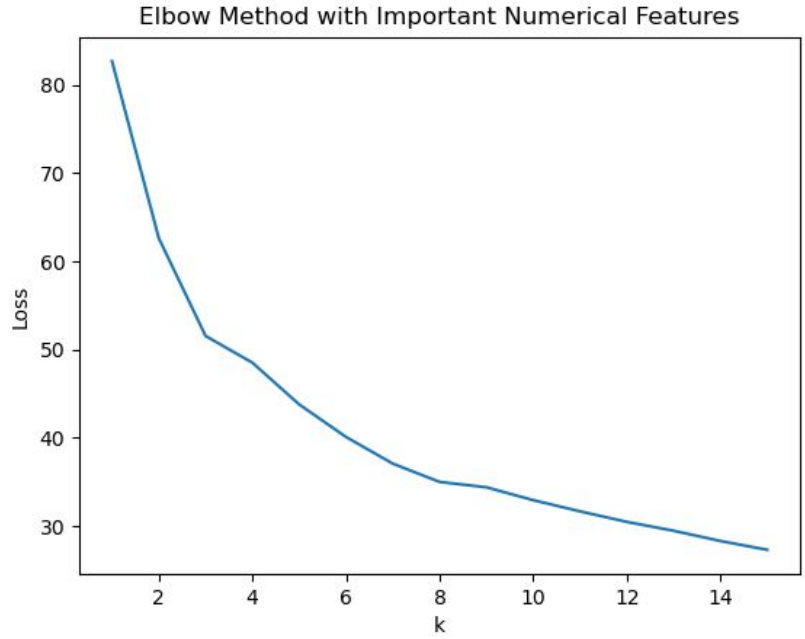

Figure 5. Graph of loss with respect to number of clusters for important features.

Using the elbow method heuristic to choose our number of clusters for our model, we choose k=5 for the model of all numerical features and k=6 for the model of limited features.

The resultant models for the unsupervised approach benefited from fast training times due to the PCA reduction for the numerical features. Because no genre labels were used, the models did not suffer from the label bias towards Rock. The chosen cluster numbers for the models were both less than the total number of genres represented in the dataset, so the clusterings must contain a variety of genres. However, the models' method for selecting the cluster count is lacking, as the elbow, in many of these graphs, is not obvious. K-Means clustering also comes with its inherent assumption that the clusterings are disjoint, which will limit the diversity of songs the user is given.

Supervised Approach

The supervised approach was implemented in scikit using minimal cost-complexity pruning decision trees with the label for the data being genre ("Post Pruning Decision Trees with Cost Complexity Pruning."). The algorithm for minimal cost-complexity pruning adds a parameter $\alpha \geq 0$ to create a complexity parameter defined as $R_\alpha(T) = R(T) + \alpha|T|$. |T| is the number of leaf nodes in the tree T, and R(T) is the misclassification rate. $R_\alpha(T)$ is minimized, and different values of α are evaluated for on training and testing sets ("1.10. Decision Trees."). The process is described for our specific dataset in more detail in the next paragraph.

The first step is generating the α's from 0 to the α that creates an arbitrary decision tree of 1 node; these α's are then used to train decision trees. The data was split into 75% training and 25% testing. Below is the graph showing as the α increases, total node impurity increases because the tree is being pruned.

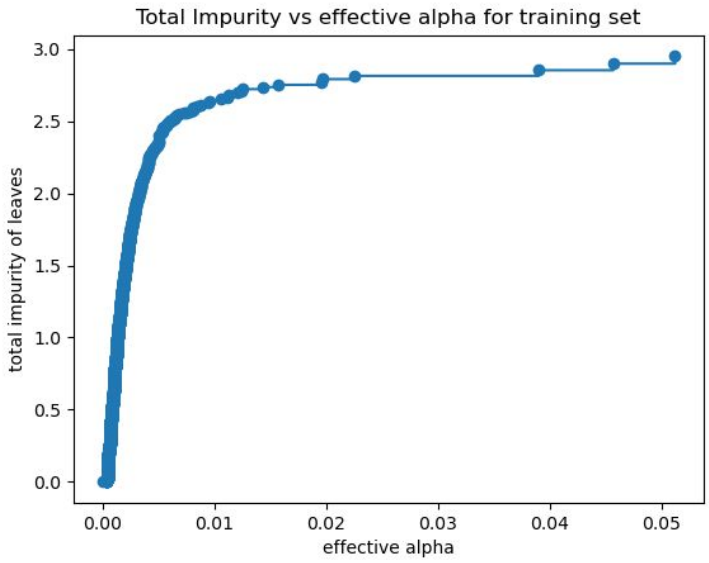

Figure 6. This graph shows that as α increases, the purity of nodes decrease; this makes sense because as the penalty for larger trees grows, the tree removes more links.

The evaluation of accuracy on the training and testing sets for different α's is shown below.

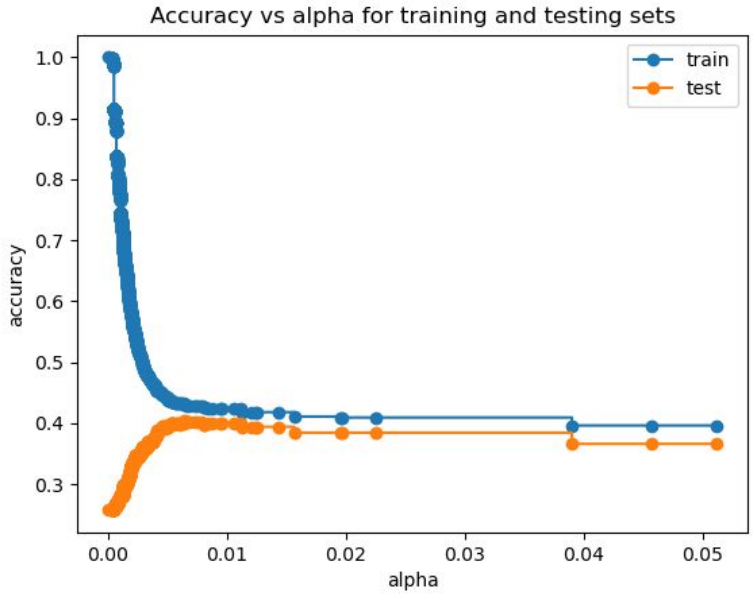

Figure 7. Graph of accuracy on the training and testing sets for all α's.

The most effective alpha of $\alpha = 0.0064$ was taken from this graph to train the final decision tree. The decision tree is visualized below.

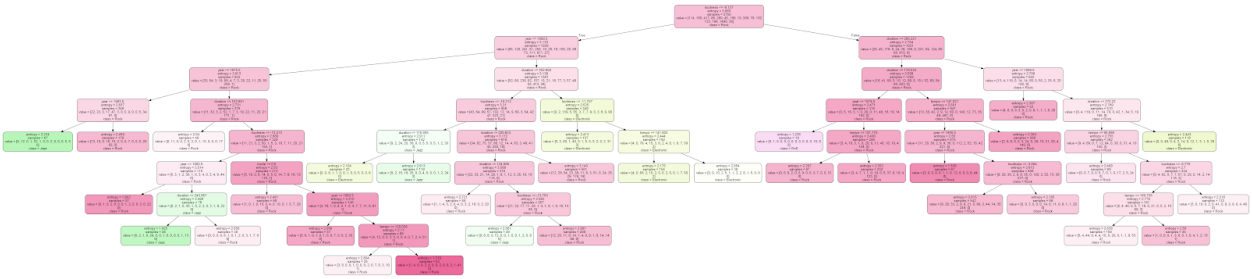

Figure 8. The decision tree trained with the optimal α.

The decision tree has a max depth of 9, and the colors show the purity and predominant genre of the node.

The results of the decision tree lead a lot to be improved. The good attributes of this method is that it yielded a simple model with easy visualization and a . The downsides are that this model is heavily biased towards Rock since that was the plurality of labels and the accuracy of the training set was only ~40%. Similar results were found when pre-processing the data using PCA as shown below.

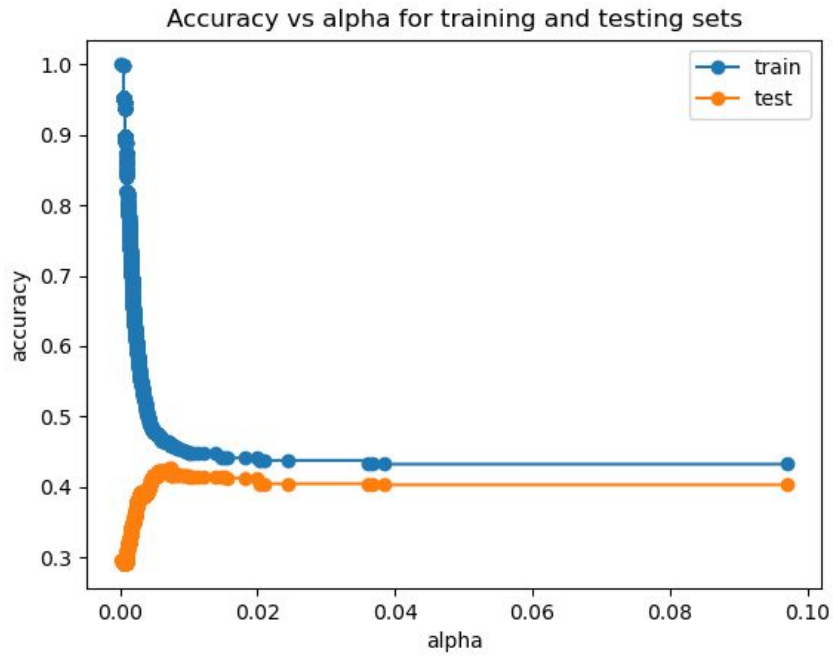

Figure 9. The same graph as in figure 7, but this is for data pre-processed with PCA.

Discussion

The summary of the unsupervised and supervised results are summarized in the table below:

Approach	The Good	The Bad
Unsupervised	● Fast evaluation ● Little bias ● No genre constraint	● Disjoint song groupings ● Ambiguous cluster count ● Difficult visualization
Supervised	● Simple solution ● Good visualization	● Inefficient implementation ● Heavily biased

The unsupervised method's positives were: fast evaluation, little bias, and no constraint due to genre; however, the song groupings were disjoint, the optimal number of clusters was ambiguous, and visualization of the model was difficult. For the supervised method, the plusses were that it yielded a simple solution with strong visualization, but the implementation was inefficient because it trained a decision tree for every α and the results were heavily skewed towards the predominant genre of rock. Both strategies have much room for improvement as discussed in future work.

Future Work/Improvements

For future work, we believe that obtaining a more balanced dataset would be a good first step. Eliminating the bias toward Rock songs in the decision tree could improve the overall accuracy of this supervised model. Further analysis of the unsupervised approach through tweaking the model parameters may also help yield a well-defined elbow graph.

For the unsupervised model, a richer set of song features may help in creating more robust clusters. To remedy the disjoint assumption, a Gaussian Mixture Model may be used instead, to probabilistically map the songs to a centroid. Furthermore, the unsupervised model could be integrated into the supervised model by training the decision trees off of the cluster labels instead of the genre labels.

While these two approaches have great potential to solve this problem, it may also be worth investigating some alternative reinforcement learning models in future developments. One future approach could be converting the song into a spectrogram and analyzing it with a deep convolutional neural network. It should be clear that many approaches exist for song genre classification and analysis.

Citations

"1.10. Decision Trees." *Scikit*, scikit-learn.org/stable/modules/tree.html.

Barnwell, Aliya. "The 5 Best Exercise Music Apps." *Digital Trends*, Digital Trends, 28 June 2015, www.digitaltrends.com/health-fitness/best-exercise-music-app/.

"Liquidata / Million-Songs." *Dolthub.com*, 15 Apr. 2020, www.dolthub.com/repositories/Liquidata/million-songs.

Luff, Christine. "How to do the Run-Walk Method." *Verywell Fit*, 9 Mar. 2020, www.verywellfit.com/how-to-do-the-runwalk-method-2911203.

Markell, Jenny. "Can Listening to Music Improve Your Workout?" *National Center for Health Research*, www.center4research.org/can-listening-music-improve-workout/.

"Million Song Sample Dataset." *Aws.amazon.com*, 2014, aws.amazon.com/datasets/million-song-sample-dataset/.

Nasrullah, Zain, and Yue Zhao. "Music Artist Classification with Convolutional Recurrent Neural Networks." *2019 International Joint Conference on Neural Networks (IJCNN)*, 14 Mar. 2019, doi:10.1109/ijcnn.2019.8851988.

"Post Pruning Decision Trees with Cost Complexity Pruning." *Scikit*, scikit-learn.org/stable/auto_examples/tree/plot_cost_complexity_pruning.html.

Schreiber, Hendrik, and Meinard Muller. "A SINGLE-STEP APPROACH TO MUSICAL TEMPO ESTIMATION USING A CONVOLUTIONAL NEURAL NETWORK." *Proceedings of the 19th ISMIR Conference, Paris, France*, 23 Sept. 2018, pp. 23–27.

Schreiber, Hendrik. "Tagtraum Genre Annotations for the Million Song Dataset." *Tagtraum Industries*, 2015, www.tagtraum.com/msd_genre_datasets.html.

"Sklearn.cluster.KMeans." *Scikit*, scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html.

"Sklearn.decomposition.PCA." *Scikit*, scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html.