**Week 4 Writeup - Make the Data Model Ready**

**Ryan O’Hara, Jack Metzger**

**Data Cleaning**

Much of our group’s data cleaning was done in weeks 2 and 3.

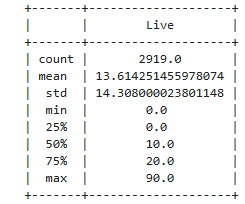
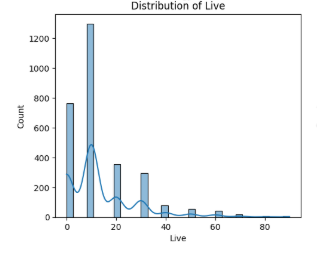
Week 2 involved adjusting the dataset to draw from 17 public playlists on Spotify (8,600 songs). The first step was removing all duplicates and remixes. Then genre data was scraped for 5,500 of the songs from [musicbrainz.org](http://musicbrainz.org). Lyrics were scraped for 4,800 of the songs and non English songs were removed leaving 4,488 songs. Lastly, songs with slightly different names, remastered, or included features by other artists were removed from the dataset, leaving 4,095 songs.

In week 3, more cleaning was done to refine the dataset. First, redundant columns were dropped. Next, all songs with a popularity score of less than 20 were dropped from the dataset. Lastly, each song’s genres were trimmed down from 5 listed genres to one primary genre and a subgenre using a systematic method.

**Week 4 Data Cleaning**

*Dropping ‘Live’ and ‘Time’*

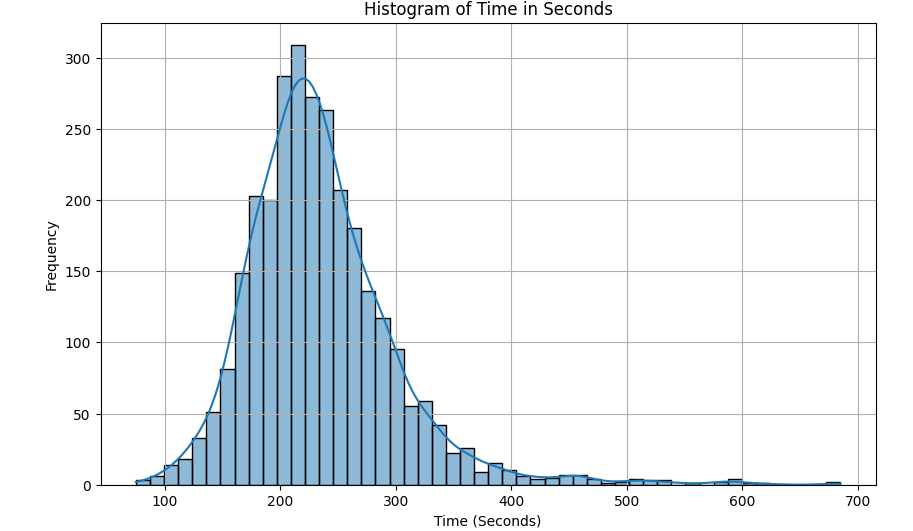
Live: The column ‘Live’ was dropped from the dataset. ‘Live’ is defined in the dataset as indicating the presence of an audience in the recording. ‘Live’ is scored on a scale from 0-100 with a score of 80+ strongly suggesting a live track.



**Figure 1: (Left):** Histogram depicting the distribution of the ‘Live’ feature within the dataset. Note the unimodal (at 10) distribution with a heavy right skew. **(Right):** Describing the ‘Live’ column. Note the low mean value of 13.614 and IQR of 0-20.

With the low mean value of 13.614, and IQR of 0-20, and a standard deviation of 14.3 we can see that 95% of the data falls between 0 and 41.66, alluding to a dataset heavily biased by non-live music. This makes sense as most music found on streaming platforms are recorded in studio, with a minority of songs having live versions. Our decision to remove this column from the dataset stemmed from this imbalance. We considered turning this feature into a binary variable, but feared the imbalance of live and recorded songs had potential to bias our model and only recommend live songs when the model is fed a live song to recommend based off of and vice versa. Another thought was to find more live recorded songs for the dataset in an attempt to make it more balanced, however this would artificially bias the model away from the real world distribution of live vs recorded songs on streaming platforms. We felt keeping the integrity of real world distributions of the Spotify library was important to maintain the integrity of the dataset and the random sampling done to obtain it. Lastly, our group felt that upon all the metrics you can list, a song being live or recorded had little significance on deciding if a person liked or disliked a song. Therefore, the best decision is to drop the column from the dataset.

Time: The ‘time’ feature represents the length of the song seconds. ‘Time’ was also binned into time buckets, spaced out by 10 seconds. The driver behind removing ‘time’ from the dataset is we feel as this is not a relevant metric for deciding if someone will like a song or not.



**Figure 2:** Histogram depicting the distribution of the ‘Time’ feature. Note the normal distribution with a right skew.

The data distribution itself was workable. By normalizing the feature, it would have worked in the model. However, we are trying to find features of songs that would drive someone to like or dislike the song such as BPM, Dance, Energy, Happy, Loud, and Lyrics. The concern was the model having the potential to only recommend songs of similar song length, taking away the impact of some other features that may be able to better predict someone’s preference for a song.

*Genres:* When benchmarking our recommendation algorithm against the classifications (genres) given by our datasource, we needed to make sure we had proper genre labels for each song. Our original dataset that we collected supplied five “genre tags” for each song, so narrowing these tags into one single genre / subgenre was very important.

In order to do this, we analyzed the most prevalent genres present in the dataset and found there to be 11 prominent main genres and 17 possible subgenres. The first main genre that appeared in the list of genre tags would be the songs genre, and then we would go back to the front of the list and the first subgenre (that wasn’t the main genre) would be added as the subgenre. This categorized songs into 11 main genres and over 150 possible subgenres.

To get these subgenres down to a manageable amount for classification, any subgenre that had less than 15 songs in it would be reassigned to simply its main genre. For example, there were only 3 songs classified as “indie synth pop” so they were converted to simply “indie”. This brought down the amount of total subgenres to just 42, much easier to work with.

**Standardizing/Normalizing the Data:**

Normalizing data is an important aspect of the data preprocessing before employing a model. “Normalization is a specific form of feature scaling that transforms the range of features to a standard scale. Normalization and, for that matter, any data scaling technique is required only when your dataset has features of varying ranges… Normalized data enhances model performance and improves accuracy of a model. It aids algorithms … by preventing features with larger scales from dominating the learning process” (Sejal Jaiswal, [datacamp.com](http://datacamp.com)).

Our method of normalizing is MinMax scaling. This method defines the minimum feature equal to zero and the maximum feature equal to one ([geekforgeeks.com](http://geekforgeeks.com)). Put differently, sklearn’s MinMax scaler can be used to shrink the data within the given range of 0 to 1. It is important to note that utilizing the MinMax scaling method does not change the shape of the distribution.

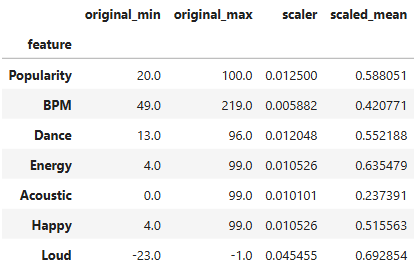
After dropping the ‘Live’ and ‘Time’ from the training dataset, all remaining numeric features were normalized to fit a scale from 0 to 1 using sklearn’s MinMaxScaler function. Those features included ‘popularity’, ‘BPM’ (beats per minute), ‘dance’, ‘energy’, ‘acoustic’, ‘happy’, and ‘loud’.

The features ‘popularity’, ‘dance’, ‘energy’, ‘acoustic’, and ‘happy’ followed a scale of 0 to 100 before normalization. The normalized values for these columns remain more interpretable in relation to the original values than the features that did not follow a 0 to 100 scale. However, we cannot assume the normalized values represent the original value divided by 100 because none of the features contained a minimum value of 0 and a maximum value of 100 in the original distributions.

*BMP:* BPM’s original distribution contained a minimum of 49 and a maximum of 219. Without normalizing, the model was at risk of weight bias, weighing BPM as a more significant metric due to its higher value distribution. This is especially true in deep learning neural networks which use a weighting system on the parameters.

*Loud:* Loud is measured in decibels. The original distribution contained a minimum of -26 and a maximum of -1. Without normalizing, the model was at risk of a number of biases. The scale imbalance would have the potential to negatively impact performance on deep learning and distance models. With one negative feature, any model utilizing a gradient descent optimization algorithm would be at risk of unstable weights and weight updates. In distance models such as K-neighbors, a negative metric can dominate the distance metric and dominate the model ([scikit-learn.org](http://scikit-learn.org)).

By normalizing all the features, we can be confident the model will weigh all the features will equal bias and on the same scale. This will aid model performance and interpretability of the data. Rather than certain features such as loud following an arbitrary scale they now lie on a scale of 0 to 1, with 0 being the recorded minimum for that feature in the training set and 1 being the recorded maximum.



**Table 1:** Table showing each numeric feature’s original min and max, the scaler used to scale the feature, and the scaled mean of the feature.

Scaler = 1/ (max - min).

**Preventing Data Leakage** Our team took intentional measures to prevent data leakage throughout the preprocessing and modeling pipeline, ensuring the integrity of our model and its ability to generalize to new, unseen data. When information from outside the training dataset is inadvertently incorporated into the model-building process. We called this process "data leakage”, which usually occurs due to the optimistic performance projections leaning on results that lead us to poor real-world applicability.

First, we split the dataset into training, validation, and test sets before implementing any of the normalization or standardization processes. This choice was essential to preserving the test set's independence. Only the training data was utilized to determine the normalization parameters, such as the lowest and maximum values for each feature used in MinMax scaling, which were subsequently applied to the test data. This method made sure that the training process didn't contain any test-set statistical information.

Additionally, we were cautious when adding characteristics that might act as stand-ins for the target variable. One example of this is the "popularity" score. We knew that keeping this feature would lead to suggestion results and user behavior once it was available. We used feature importance analysis to evaluate the influence of "popularity," treating it as an extra signal rather than a dominant predictor, to reduce this danger.

Furthermore, we refined the genre metadata to avoid semantic leakage. We eliminated duplicate or genre tags, leaving only primary and subgenres. This process helped guarantee that tightly defined categories did not inflate model performance and prevented overfitting of genre-specific clusters.

Finally, we eliminated variables such as playlist co-occurrence, user ratings, and collaborative filtering artifacts that could be obtained from downstream interactions. Every element that was kept was essential to the song. For instance, by scraping external databases, variables like tempo, danceability, energy, lyrics, and other characteristics are independently acquired. We were able to keep training input and any possible feedback loops separated by concentrating on static, pre-consumption features.

By following these procedures, we reduced bias, maintained the temporal and structural integrity of our data, and increased the likelihood that our model would remain impartial, understandable, and helpful in practical applications.

Works Cited

“7.3. Preprocessing Data.” *Scikit*, scikit-learn.org/stable/modules/preprocessing.html. Accessed 8 June 2025.

Hemavatisabu. “Data Pre-Processing with Sklearn Using Standard and Minmax Scaler.” *GeeksforGeeks*, 3 Feb. 2022, www.geeksforgeeks.org/data-pre-processing-wit-sklearn-using-standard-and-minmax-scaler/.

Jaiswal, Sejal. “What Is Normalization in Machine Learning? A Comprehensive Guide to Data Rescaling.” *DataCamp*, DataCamp, 4 Jan. 2024, www.datacamp.com/tutorial/normalization-in-machine-learning.