**Week 5 Writeup - Feature engineering, Data augmentation, Dimensionality reduction**

**Ryan O’Hara, Jack Metzger**

**Tokenization + Updates to W4 Notebook**

This week we tokenized and vectorized the lyric data so it is ready to be used for our latent space in the coming weeks. We noticed that some of the contractions that were present in the lyrics before cleaning remained in them post cleaning, because of this one of the steps we had to do this week was to go back and update our week 4 notebook to include contraction cleaning before using the spacy package to clean. This changed the contractions for example from can’t to cannot - improving the retention of semantic meaning in the lyrics.

The next step was to tokenize the song lyrics using NLTK, which involved breaking down the raw text into individual tokens (words and punctuation). This step is important for cleaning and preparing the text data for further analysis, helping to standardize the input and reduce noise.

After tokenization, vector embeddings were generated for each token using FastText. Different packages were researched and considered but in the end due to the song sample size and the semantic meaning retention, FastText was used. FastText represents words as a combination of character n-grams, allowing the model to capture subword information and better handle rare or misspelled words common in song lyrics. This method creates dense vector representations that encode semantic similarities and relationships between words which is key for the goal of our recommendation model. Moving forward, these vector representations will be integrated into the recommendation pipeline to improve song similarity measures and enhance user personalization.

\**In order to run the week 5 notebook smoothly, the embedding from FastText must be downloaded and the path converted to that of your own. The link and name of the file is included in the notebook - could not upload to github due to size of file.*

**Feature Engineering**

“Feature engineering is the process of transforming raw data into features that are suitable for machine learning models… It is the process of selecting, extracting, and transforming the most relevant features from the available data to build more accurate and efficient machine learning models ([geeksforgeeks.org](http://geeksforgeeks.org)).

Previously Completed Feature Engineering Steps:

* Genre Cleaning/Engineering
* Data Normalization

*One-Hot Encoding/Label Encoding: Camelot, Genre, Subgenre:*

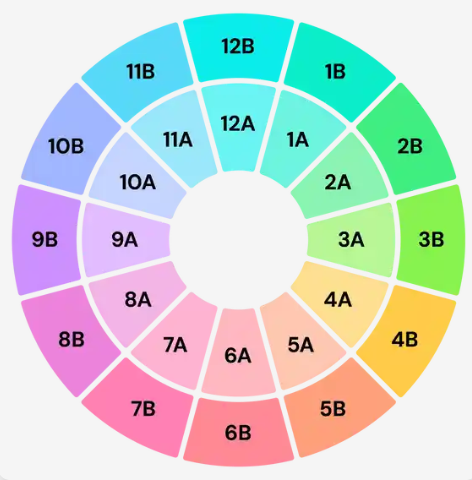
Categorical variables are defined as “data [that] represents discrete values or labels that fall into distinct categories or groups” ([medium.com](http://medium.com), Chandima Jayamina). Categorical variables can be nominal, ordinal, or binary. Nominal variables have no rank or order (genre/subgenres). Ordinal variables have an ordered structure (camelot). Binary variables contain two states (yes or no/ present or not present). Traditional machine learning methods require numerical data, making categorical variables in their raw form unsuitable for ML models. The categorical variables in our dataset, ‘camelot’, ‘genre’, and ‘subgenre’, needed to be encoded in order to be used in a machine learning model. The concept of one-hot encoding works to transform categorical data into multidimensional binary vectors, enabling the features to be used in mathematical machine learning models.

*Genre and Subgenre (Nominal)* - Genre and subgenre were one-hot encoded using standard one-hot encoding. The goal was to transform each unique genre or subgenre into a separate binary column. Sklearn’s OneHotEncoder package was used to achieve this.

After applying one-hot encoding, each of the 11 primary genres (such as rock, pop, hip-hop, etc.) and their 42 associated subgenres are represented as separate binary columns in the dataset. To ensure statistical reliability and balanced representation across categories, each genre and subgenre included has a minimum of 15 songs.

*Camelot* - Camelot is defined as “a term that refers to the concept of key signatures, chord progressions, and harmonic relationships between notes and chords… In essence, Camelot is a mathematical concept that helps musicians understand the relationships between different keys, chords, and scales. It’s based on the idea that certain notes and chords have a special affinity with each other” ([clrn.org](http://clrn.org)).

The Camelot Wheel is a tool for DJs to understand and apply harmonic mixing (dj.studio). Our data labels camelot on a key scale of 1-12 paired with a mode (A or B), giving 24 total combinations. Although ordinal, camelot follows a circular scale, not a linear one. This means key 12B is adjacent to 1A. Using a linear ordinal system, the distance between 12B and 1A would be large where in reality they are just as close as 1A and 2A.



**Figure 1:** The Camelot Wheel. (image sourced from dj.studio).

To solve this problem, we used circular encoding for the camelot feature. This treats camelot like a wheel or unit circle. This will be important when modeling our songs in latent space, as neural networks and latent space models are able to see the true relationships of the camelot wheel. The resulting features from this encoding strategy are ‘camelot\_sin’ and ‘camelot\_cos’, representing the circular position or coordinates on the unit circle (camelot wheel in this case).

In addition to correctly depicting the true camelot relationships in our data, this circular encoding also cut down on data dimensionality. Traditional one-hot-encoding would have added 24 dimensions to the data (one column for each label). Circular encoding results in adding only two dimensions to the data (columns comelot\_sin and camelot\_cos).

*Lyric Tokenization:*

We included a tokenization phase that converts cleaned lyrics into lists of individual words, allowing us to incorporate lyrical content as a feature. We proceeded with tokenization because it is an important part of NLP and converts unprocessed text into a format suitable for vectorization and modeling.

We defined the function called simple\_tokenize, which splits each lyric string using the split function. Although more sophisticated tokenizers such as NLTK’s word\_tokenize could be applied to more intricate linguistic details, the more straightforward approach proved more effective and adequate in this situation, given the size of our dataset and the emphasis on downstream embedding.

The tokenization step was applied to the clean\_lyrics column in the dataset, which has been split into the train, test, and validation sets. Tokens are created in the new Tokens column in datasets.

After tokenization, FastText was used to get these tokens ready for embedding. We obtained FastText English word vectors that had already been trained, which is called crawl-300d-2M-subword, and then they transformed each token list into a fixed-size numerical vector using them.

Each song's embedding was calculated using the mean of the FastText vocabulary's token embeddings. A zero vector was given to songs whose lyrics exclusively contained tokens that were not in the song's lexicon.

The lyrical content can now be mathematically represented in a way that captures the semantic links between words, thanks to this embedding phase. Now there is the column called Embeddings, which is ready for incorporation into subsequent models, including clustering tasks or recommender systems.

*Interaction Term: Energy and Loud:*

We included both Energy and Loud as normalized individual features in our dataset, each ranging from 0 to 1. While both variables are related to the acoustic intensity, they captured some characteristics, such as energy, which measures the dynamism, rhythm, and overall activity, and loud measures volume. The two demonstrated a noteworthy correlation coefficient of 0.73, indicating a substantial link, despite this conceptual distinction.

There is a new variable called "power," which is the multiplication of energy and loud. This makes it easy for us to capture the relationship between two variables. For example, genres such as heavy rock or dance show a higher value of power.  
  
 This process has been preceded by both data splitting and normalization to maintain the original scale of the feature and guarantee that the interaction represented real audio dynamics instead of values that had been post-processed. Our intention to preserve significant distributions and enable the model to represent genuinely occurring interaction patterns served as the foundation for this choice.

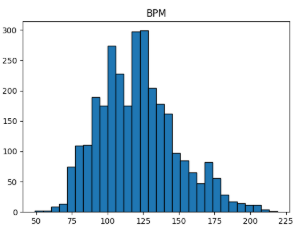
Even though the paper's initial draft claimed that no interaction terms were used, that has since been fixed. The Power variable can provide accuracy for both experimental and acoustic music analyses.

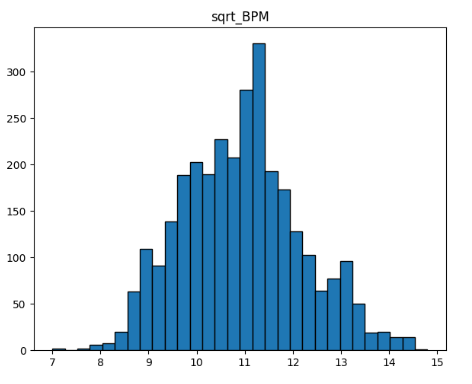
*Data Transformation/Distribution Normalization:*

Many of our features naturally follow a normal distribution with minor skew, making them suitable for a machine or deep learning model. We were able to create a more normal distribution for two of the features with basic transformation functions: BMP and Acoustic. This step was completed before splitting and normalizing the data to fit a 0 to 1 scale.

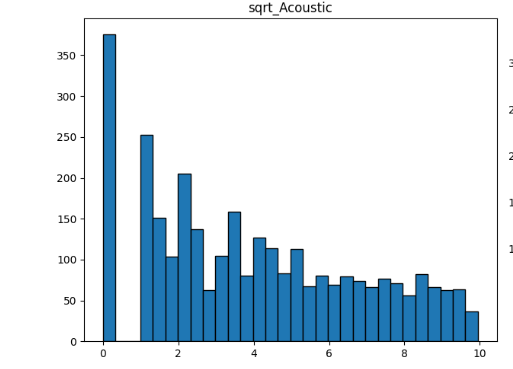
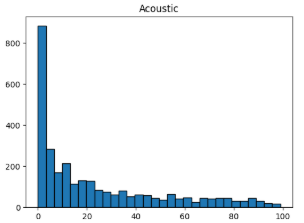
*BPM:* The original distribution of BPM follows a normal distribution with a right skew. To further normalize this feature the square root was taken.

*Acoustic:* The original distribution of Acoustic is extremely right skewed. Both log transformation, square root, and cube root transformations were attempted with the best results coming from the square root transformation. This did not completely solve the acoustic distribution problem, but it helped create a more spread distribution for the model.

*Other Features:* We decided not to transform other features of the data. Many features follow a very minor left skew. Left skewed data is harder to fix with simple transformations and the original feature interpretability outweighed any minor benefit to the distributions.



**Figure 2:** Original BPM distribution (left). Transformed BPM distribution - square root transformation (right).



**Figure 3:** Original acoustic distribution (left). Notice the heavy right skew. Transformed acoustic distribution (square root) (right).

**Dimensionality Reduction**

Since we are building a recommendation system that relies on capturing detailed relationships, we chose not to reduce dimensionality at this stage in order to preserve as much relevant information as possible. When we iterate to improve our model we may try reducing the dimensions to test if that leads to better accuracy but for now we believe the more information the better.

Our decision to use circular encoding, utilizing sine and cosine coordinates to represent the Camelot Wheel helped keep dimensions down. The resulting dimensions from circular encoding was two vs twenty-four using traditional one-hot-encoding.

Works Cited

DJ.Studio. “The Camelot Wheel for Djs: Explained.” *DJ.Studio | The Camelot Wheel for DJs: Explained*, dj.studio/blog/camelot-wheel. Accessed 15 June 2025.

Jayamina, Chandima. “ML: How to Deal with Categorical Data and What Is Dummy Variable Trap.” *Medium*, Medium, 21 Aug. 2024, chandimajayamina.medium.com/ml-how-to-deal-with-categorical-data-60ea4c40709b.

team, CLRN. “What Is Camelot in Music?” *California Learning Resource Network*, 27 Dec. 2024, www.clrn.org/what-is-camelot-in-music/.

“What Is Feature Engineering?” *GeeksforGeeks*, GeeksforGeeks, 21 Dec. 2023, www.geeksforgeeks.org/machine-learning/what-is-feature-engineering/.