MIE368 - Analytics in Action:

Project Final Report

## Content-Based Spotify™ Playlist Recommendation System

November 27, 2024

*Team 20*

### Hussein Ismail (1008747910)

### Marco Mastrangelo (1009133518)

### Devraj Solanki (1009065707)

### Yahia Hussein (1008956444)

**1.0 Motivation:**Providing good song recommendations to users is a challenge that many music streaming platforms face [1]. On popular streaming platforms such as Spotify, collaborative filtering techniques are mainly used to recommend new songs to users [2], where recommendations made for a user are based on a similar user’s likings. However *this approach falls short in providing highly personalized suggestions*.To address this,*the team proposes a content-based recommendation system* [3] *that suggests new songs to add to a playlist based on its existing content*. This allows the team to provide curated recommendations for a specific playlist rather than giving more general recommendations based on other user’s listened-to songs.

**2.0 Data Collection and Cleaning:**

To collect music data and information, the team queried the Spotify Web API [4] to scrape data from 22081 unique songs. To retrieve these songs, a dataset of the top 2500 most streamed Spotify artists was utilised [5]. For each of these artists, the artist’s top 10 most streamed tracks were retrieved, resulting in a total of 25000 songs. However, because of collaborations between artists, multiple artists shared top 10 tracks. Therefore, these duplicated tracks were removed from the dataset, resulting in 22081 songs. General information about the songs such as the track name, artist name(s), and release date were retrieved. Some relevant audial metrics were also retrieved for each song including ‘instrumentalness’, ‘acousticness’, ‘speechiness’, ‘danceability’, ‘valence’ (all numeric values, ranging from 0 to 1) , ‘popularity’ (ranging from 0 to 100), and ‘tempo’ (measured in BPM)[4]. Some other metrics that were scraped were deemed to be irrelevant for the project, such as 'uri', 'id', 'track\_href', 'analysis\_url'.

To collect playlist data the Spotify API again is queried. 10 playlists were collected that were either genre-based like Rock and Rap playlists, or activity-based like Workout or Study playlists; for each song in the playlist, relevant metrics were also collected, just as they were with the 22,081 songs. To transform the playlist into a dataset that can be used to train a classification model, the songs from the playlist are combined with an equal number of randomly selected songs. A target column is then created where a class = ‘1’ signifies that the song is in the playlist and class = ‘0’ means the song was randomly selected.

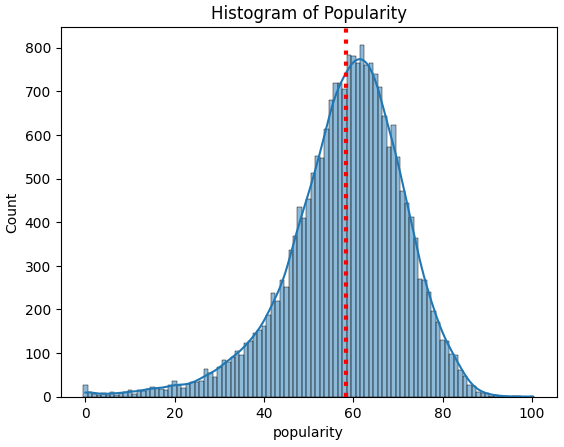
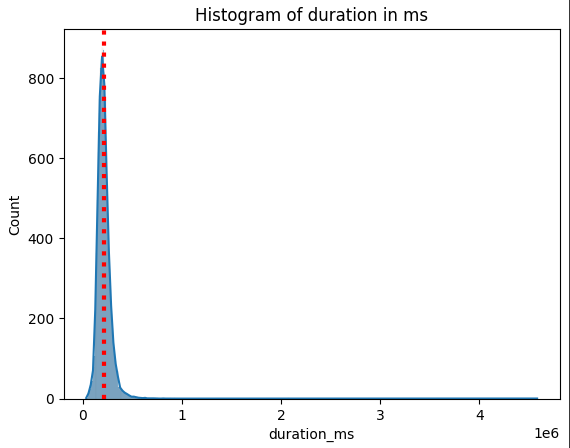
To clean the data, routine checks were carried out, such as confirming data types and removing remaining duplicates. Furthermore, after checking for missing data, the team discovered 4 null values in the ‘release date’ column. The team imputed by cross referencing the ‘tracks names’ column with song names in the music database called *discogs* [7] to find their release date. After printing the release date column, the team noticed that the formats were either a full date formatted as (year/month/day) or just a year. In order to ensure consistency, the team truncated the release date column to only include the release year for all tracks.

2.1 EDA:

To perform EDA, the distributions of each feature used were first visualized.

The histogram for popularity score showed an approximately normal distribution with a mean of approximately 60 with a slight left skew. This left skew is expected as the data was generated from top artists, therefore most tracks should have a high popularity score see figure 1.

The histogram of track duration seemed to be normally distributed with a mean of approximately 3.3 minutes. The plot also exhibited a strong right skew see figure 1.



*Figure 1: Example of histograms used for EDA.*

The team used a subset of audial features to generate violin plots, in order to visualise how they are distributed. The features included 'danceability', 'energy', 'acousticness', 'speechiness', 'instrumentalness', 'liveness', 'valence', 'tempo', and 'loudness'.

Based on the visualisations, it seems that certain song metric features such as valence, energy, danceability and acousticness capture a greater variability of values, as seen in Figure 2(a). Therefore, these features could be important when explaining the variability of the data (and will be considered during model creation). Other features such as instrumentalness and speechness capture less variability as seen in Figure 2(b). The features may not be as important in explaining the variability of data.

|  |  |
| --- | --- |

*Figure 2: Violin Plots to visualize distribution of (a) Valence and (b) Instrumentalness.*

Furthermore, from the violin and histogram plots, it is clear that many of the features are either non-normally distributed or skewed. This would impact the types of models we can use on our data. Specifically, models that assume normally distributed, (ideally) non-skewed variables such as linear models (i.e. Linear Regression) can not be used. Additionally, highly skewed variables can impact distance calculations, therefore before using ML models that use distance metrics, data should be log transformed inorder to account for the skewed data.

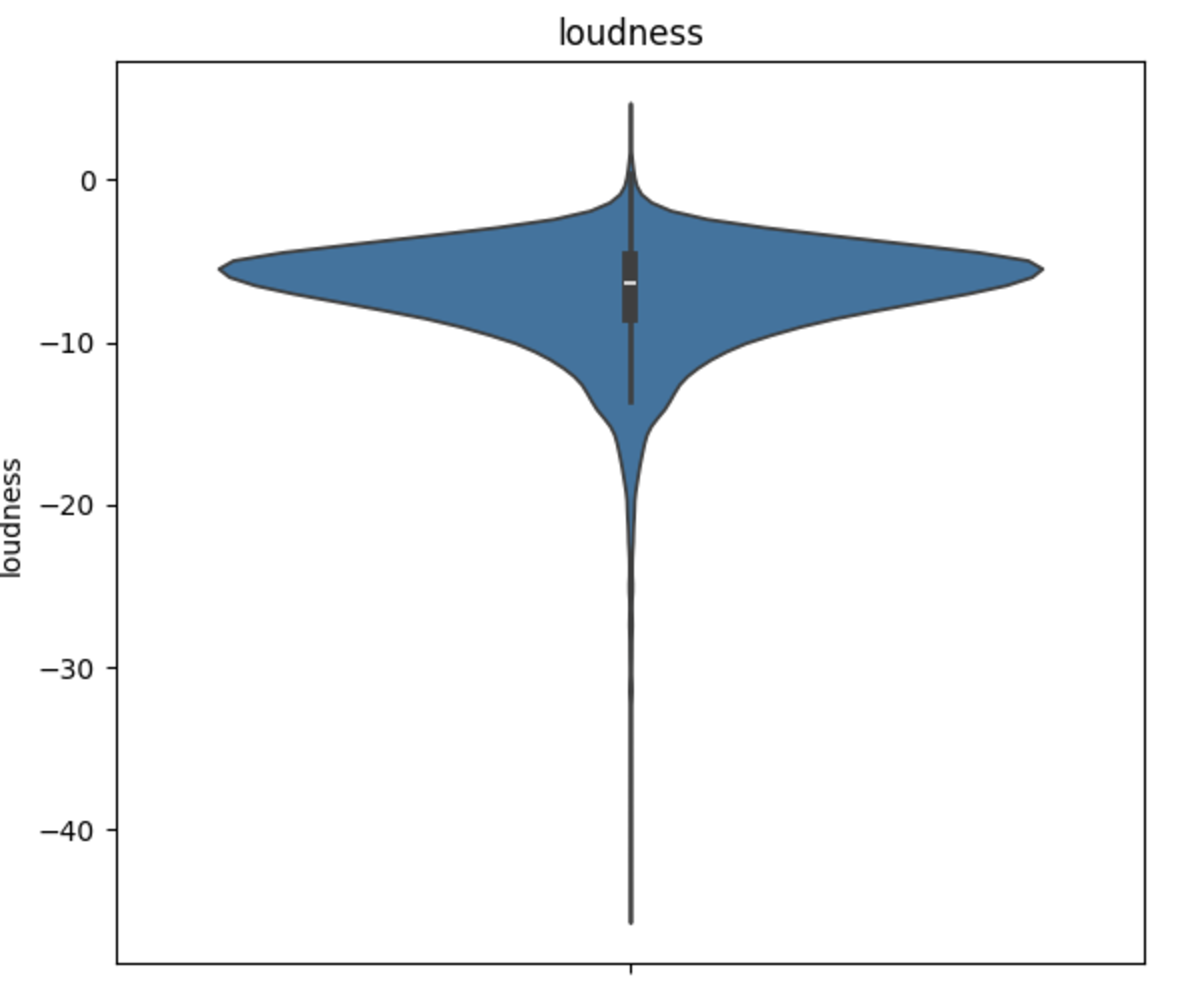
In order to understand which features might be collinear, the team created a heatmap see Appendix A. From the correlation heatmap, it seems that loudness and energy are highly positively correlated (0.71) and acousticness and energy are highly negatively correlated (-0.62). Additionally, the team used a pairplot to visualise the relationships between features see Appendix B. From these pairplots, the correlation heatmap, and previously established domain knowledge, it seems that the variables: acousticness, energy and loudness are correlated with one another; and may introduce multicollinearity in our model. Therefore feature selection should be performed on these variables to remove such a possibility.

**3.0 Methods:**

The following section describes the methodologies employed to produce the final model, including techniques for feature selection, an overview of the model architecture, model training and hyperparameter tuning.

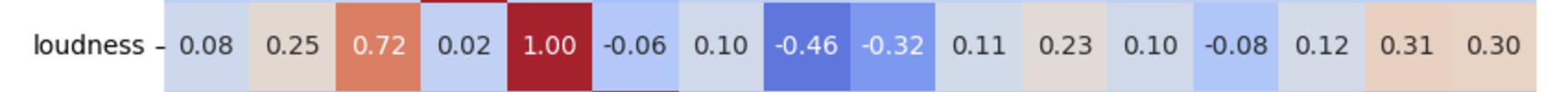
3.1 Feature Selection:

EDA uncovered that the loudness feature did not capture a significant amount of variability in the data, as seen in the violin plot for the feature in figure 3.



*Figure 3: Violin plot for loudness feature.*

It was also highly correlated with other features, namely energy (0.72) and acousticness (-0.46) as observed in the correlation plot in figure 4.



*Figure 4: Row on correlation plot for loudness compared to all other features.*

This is problematic since these features will compete for the same statistical effect, hindering the effectiveness of the model. Therefore, the team dropped the loudness feature.

3.2 Model Architecture

The model architecture was composed of both supervised and self-supervised machine learning models. A flow of the overall model architecture can be seen in Appendix C.

The model begins with a denoising autoencoder, used to generate meaningful song embeddings for each of the 22081 songs in the full dataset. Songs in each playlist are then aggregated into a “mean song” that captures the essence of the playlist; this is done by taking the average of each feature across all songs in the playlist. This “mean song” is then passed into the denoising autoencoder to generate a playlist embedding. Using cosine similarity, the top 20 most similar song embeddings to the playlist embedding are retrieved. These 20 songs are the models’ initial recommendations.

To verify the quality of these recommendations, an XGBoost classifier is trained and fit on the playlist ‘dataset’, mentioned in section 2 using RandomizedSearchCV for a total of 2500 fits. This allows the classifier to understand high-level patterns and themes, unique to each playlist. After the model is trained, tuned, and tested, the 20 recommended songs are fed into the classifier. The classifier outputs a prediction probability for each song. Probabilities closer to 1 mean the classifier has high confidence that a song belongs in the playlist, and vice versa when the returned probability is close to zero. Therefore, the 10 songs with the highest prediction probabilities with a minimum probability threshold of 0.5 are ultimately passed as final recommendations for the user’s playlist.

3.3 Semi-Supervised model: Denoising Autoencoder

A denoising autoencoder was used to capture meaningful feature embeddings for each of the 22,801 songs in the dataset. This is a crucial task; the success of the ensemble model depends on the quality of song embeddings provided by the denoising autoencoder. To ensure high quality song embeddings, a deep autoencoder architecture was employed to learn high-level features and patterns inherent in the song data. However, a deep architecture is often prone to overfitting. To ensure the autoencoder is robust and can generalise well to new song data, a regularization technique was employed. Adding noise into the input and asking the autoencoder to reconstruct the original, non-noisy input, would ensure that the model will develop a robust understanding of the song data. To do this, 2 random features from each track in the 22,081 song dataset was “masked” by setting each value to 0. The autoencoder would then be asked to reconstruct the original, “unmasked” value of each feature for each track.

*3.3.1 Autoencoder Model Architecture*

The architecture of the model consisted of 10 neural layers, 5 belonging to the encoder section and 5 belonging to the decoder section. The encoder section boasted a first layer consisting of 4096 neurons, with the next few layers containing 2048, 1024, 256 and finally 16. The decoder section used the exact same neurons per layer but in reverse order. In order to prevent overfitting, multiple regularization techniques were included in the model architecture including: Dropout, L2 regularization and learning rate scheduling. A learning rate of 0.0003 was used with a batch size of 64. The model was trained for 70 epochs.

*3.3.2 Hyperparameter Tuning for Autoencoder*

Hyperparameter tuning was performed by observing the training and validation loss at each epoch, using a predefined set of hyperparameter values.

The number of layers and neurons used in the model needed to be large enough for the model to effectively learn high-level features in the song data, however it also needed to be small enough to avoid excessive training times. The model architecture specified in 3.3.1 seemed to offer the right balance of having a low training and validation loss while maintaining a reasonable training time.

The optimal learning rate was determined by incrementally decreasing the learning rate from 0.01 down to 0.0001 and assessing the models’ training and validation loss per epoch. A learning rate of 0.0003 was determined since it obtained the lowest validation and training loss while still allowing the model to converge in a reasonable amount of time. The learning rate scheduler also assisted in adjusting the learning rate every 30 epochs to allow the model to converge to a local/global minima.

The batch size was kept at 64 because it is often an appropriate batchsize to use when training deep learning models and, more importantly, it reduced the number of hyperparameters that needed to be tuned.

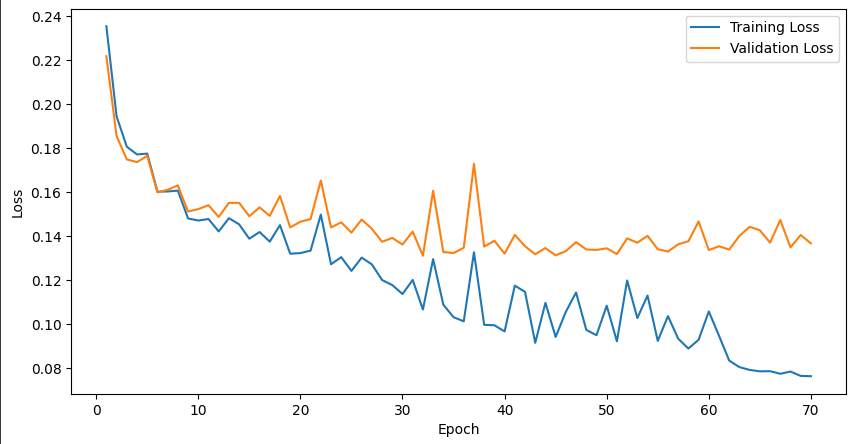
The number of epochs was incremented from 10 to 100, until the model trained long enough to show signs of overfitting. Using this methodology, an epoch count of 70 was chosen.

Various dropout probabilities (0.20,0.15,0.10, 0.05, 0.01) were used to determine the impact of dropout on training and validation loss. A dropout probability of 0.05 was chosen since it demonstrated a low enough validation and training loss while still offering regularization features.

The L2 regularization hyperparameter (λ) was chosen to be 1e-4 [6] since it is often an appropriate value to use when training machine learning models, and again, it reduced the number of hyperparameters that needed to be tuned.

*3.3.3 Results from Autoencoder*

The results from training and testing the autoencoder can be visualized through the training and testing curves below, in figure 6. The model seems to be underfitting before 20 epochs since both validation loss and training loss is decreasing, and the model seems to be overfitting after 50 epochs since the validation loss is increasing while training loss is decreasing. Therefore the optimal model state was taken at the 40th epoch where it would have a good fit. Using this epoch, the model achieved a train loss of 0.0968 and a validation loss of 0.1322.



*Figure 6: Train/validation curve for the Denoising Autoencoder on the 22081 songs.*

3.4 Supervised Model: XGBoost Classifier

To decide on which classification model to use, various machine learning models were tested including Random Forest, SVM, XGboost and Decision Trees. The team used only these models since they are able to perform well on non-linear data, which was important considering most features were non-linear (linear models such as linear or logistic regression would not perform well with non-linear data). Each of these models were trained and fit on a specially curated dataset that had a large number (1418) songs, to get an idea of which model tends to perform best.

Ultimately, XGBoost and SVMs outperformed the other models as can be seen in table 1. However, although SVMs performed similarly, XGBoost is less prone to overfitting than SVMs due to its built-in regularization techniques and ability to handle complex data relationships effectively [5]. Furthermore, as a tree ensemble method, XGBoost naturally performs feature selection, which is particularly useful in this high-dimensional space. This ensures that only the most important features are analyzed for each playlist, reducing the impact of irrelevant or redundant features. This feature selection is especially valuable as different playlists may prioritize different features—for instance, pop playlists might emphasize higher danceability scores, while rap playlists may prioritize higher energy levels, making these features more significant for effective splits.

Therefore XGBoost was selected as the team’s supervised learning model.

|  | **Decision Tree** | **Random Forest** | **SVM** | **XGBoost** |
| --- | --- | --- | --- | --- |
| **Train Error** | 0.8627 | 0.9010 | 0.9021 | 0.9031 |
| **Test Error** | 0.8517 | 0.8894 | 0.9035 | 0.9035 |

*Table 1: Training and test error for tested models*

**4.0 Results:**Due to the subjective nature of evaluating the quality of song recommendations, a mixture of quantitative and qualitative observations and metrics were used to judge the strength of recommendations generated by the team’s model.

4.1 Qualitative Results

The most straightforward way to understand if songs recommended by the model are relevant, is to cross reference these recommended songs with the songs in the provided playlist. If the recommended songs and the playlist songs share the same genre or theme, are composed by the same artists, etc. then it can be concluded that the model is capable of generating relevant song recommendations for a given playlist. An example of the models’ recommended songs for an 80s/90s rock playlist can be seen in Appendix D [7]. Evaluating these recommended songs, many of them can be recognized as rock songs from the 80’s/90’s era. Furthermore, many recommended songs were produced by artists that also have songs present in the rock playlist, such as AC/DC, Aerosmith and REO Speedwagon, further serving as validation that the recommendations are relevant. It is important to note that some of the recommendations were not completely relevant to the playlist content, such as the recommendation of songs by 2000s pop/folk rock artists such as U2, Mumford and Sons, and Beach House (these artists are not 80’s/90’s classic rock artists). This theme is prevalent among the recommendations for the other playlists as well.

An interesting observation was that recommended songs for certain playlist may align with the playlists’ theme/genre but may not be in the same *language* as songs in the playlist. An example of this can be seen in figure 7 below. The song is an Italian rap song that was recommended for a typical “American” rap playlist. While the song matches the genre of songs in the playlist, it is not a particularly relevant song to recommend due to the language differences between the song and songs in the rap playlist. This issue is derived from a flaw in the teams’ model architecture, which does not incorporate the language of songs in a playlist when generating song recommendations.



*Figure 7: Recommended song for a rap playlist.*

4.2 Quantitative Results

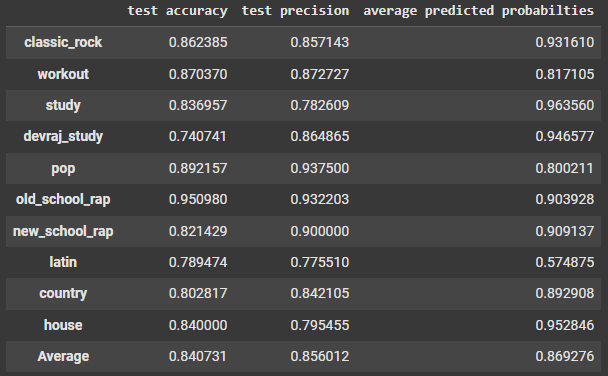
An assortment of 10 playlists were used to test the model, ranging from genre-based playlists such as rock, rap, pop and country and activity-based playlists such as study and workout.

The XGBoost model was trained on each playlist, and the predicted probability that a recommended song (provided by the autoencoder) belongs to a playlist, would indicate the quality of the song as a recommendation for that playlist. While this methodology is an ad hoc approach, given the XGBoost model may occasionally misclassify whether a song truly aligns with a given playlist, it is still a reasonable approximation of recommendation quality. This methodology is particularly relevant since song recommendations inherently lack a definitive “ground truth” label to determine the quality/fit of a song as a recommendation for a particular playlist.

Figure 8 below shows the models’ results for the 10 playlists. One thing to note is that generally, the XGBoost model displays a high accuracy and precision across all playlists. This means that the prediction probabilities produced by the supervised model are a generally reliable measure of the quality of fit of a song to a given playlist.The high precision is particularly significant because it indicates that if the supervised model classifies a song as belonging to a playlist, it has a high incidence of being correct. This enables a high confidence that recommended songs that pass the supervised model are also good quality recommendations.

From the figure, it seems that overall, the XGBoost model has a high confidence (roughly 87%) that recommended songs (provided by the autoencoder) belong to a given playlist. This means that our model is generating high quality song recommendations.

However, for certain playlists, the (average) prediction probability (across all recommended songs produced for that playlist) was much lower. For example, latin music had roughly a 57% prediction probability. This is probably due to the fact that Latin music accommodates a wide variety of musical themes and genres. For example there is Latin pop, trap, rock, folk etc. music. Playlists with a lower internal similarity (between songs in that playlist) would make it difficult to generate relevant recommendations; which is a likely reason why the Latin playlist performed poorly. This is also a likely reason that playlists like workout (which included a range of genres like pop, rock and rap) did not perform as well as other playlists.



*Figure 8: Model test results for 10 playlists; showing accuracy, precision and prediction probabilities.*

**5.0 Conclusions:**

The evaluation of the model using both qualitative and quantitative metrics provided strong evidence that content based recommendations can be a viable strategy for a recommendation system. Qualitative assessment proves the model is able to effectively learn internal song features and use them to generate recommendations. This capability is evident in the models ability to recommend songs that match playlist genres, themes, and artists. Quantitative assessment provided further evidence that the model was able to successfully generate high quality song recommendations for given playlists. This is seen through the high prediction probability achieved by the model during testing.

However, the results also highlight some limitations in the model. Notably, the model architecture does not incorporate language and regional differences when generating song recommendations. Another limitation is due to the models assumption that input playlists will have strong internal similarity. Therefore playlists with high variability between songs would not receive high quality song recommendations.

**6.0 Future Directions:**

To improve the current model and address its weaknesses, the team suggests incorporating language features to track both song language and region. Adding these features could significantly enhance the relevance of recommendations by allowing the model to better align with user preferences and cultural context. This addition is expected to improve the overall quality of recommendations and provide the model with a valuable tool for tailoring suggestions to the users preferences. Furthermore, the team could experiment with alternative aggregation methods that capture the variability of the playlist more holistically rather than relying on the mean song embedding. This experimentation is worth considering because, in practice, mean aggregating could eliminate important signals in data (which may hinder the models’ ability to provide quality song recommendations). Overall, while the model demonstrates the potential of content-based recommendation systems, addressing these limitations will be crucial for further enhancing recommendation quality and optimizing model performance.

**References:**

[1] A. M. A. Ahmed, “5 Challenges in Building Recommender Systems,” *Medium*, Nov. 04, 2023. https://awadrahman.medium.com/5-challenges-in-building-recommender-systems-77acbc0948cb

[2] N. Torabi, “The Inner Workings of Spotify’s AI-Powered Music Recommendations: How Spotify Shapes Your Playlist,” *Medium*, Aug. 28, 2023. https://neemz.medium.com/the-inner-workings-of-spotifys-ai-powered-music-recommendations-how-spotify-shapes-your-playlist-a10a9148ee8d

[3] A. Kharwal, “Content Based Filtering and Collaborative Filtering: Difference,” *The Clever Programmer*, Apr. 20, 2023. https://thecleverprogrammer.com/2023/04/20/content-based-filtering-and-collaborative-filtering-difference/#:~:text=Content%2Dbased%20filtering%20is%20like,with%20similar%20preferences%20have%20liked.

[4] Spotify, “Web API | Spotify for Developers,” *developer.spotify.com*.

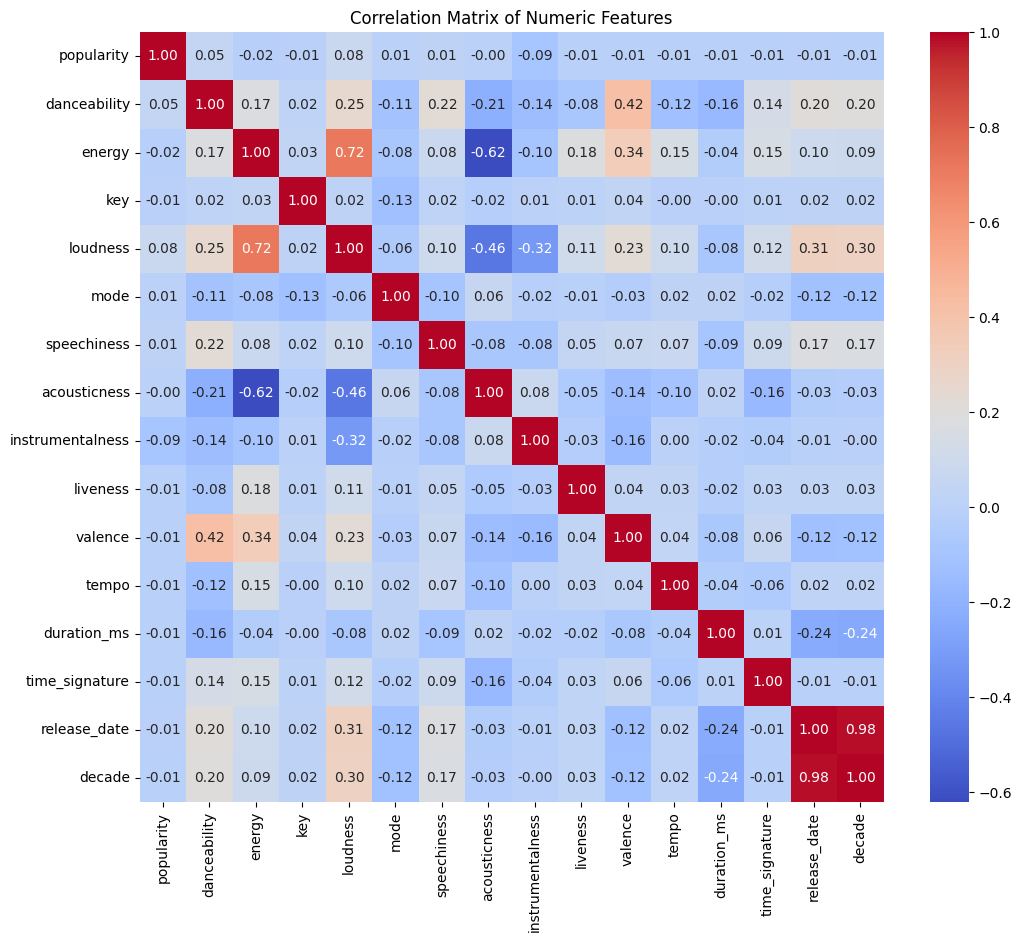
[5] “Xgboost Machine Learning | Everything You Need to Know,” *saiwa*, 2024. https://saiwa.ai/blog/xgboost-machine-learning/.

[6] “Ridge,” *scikit-learn*, 2024. https://scikit-learn.org/1.5/modules/generated/sklearn.linear\_model.Ridge.html

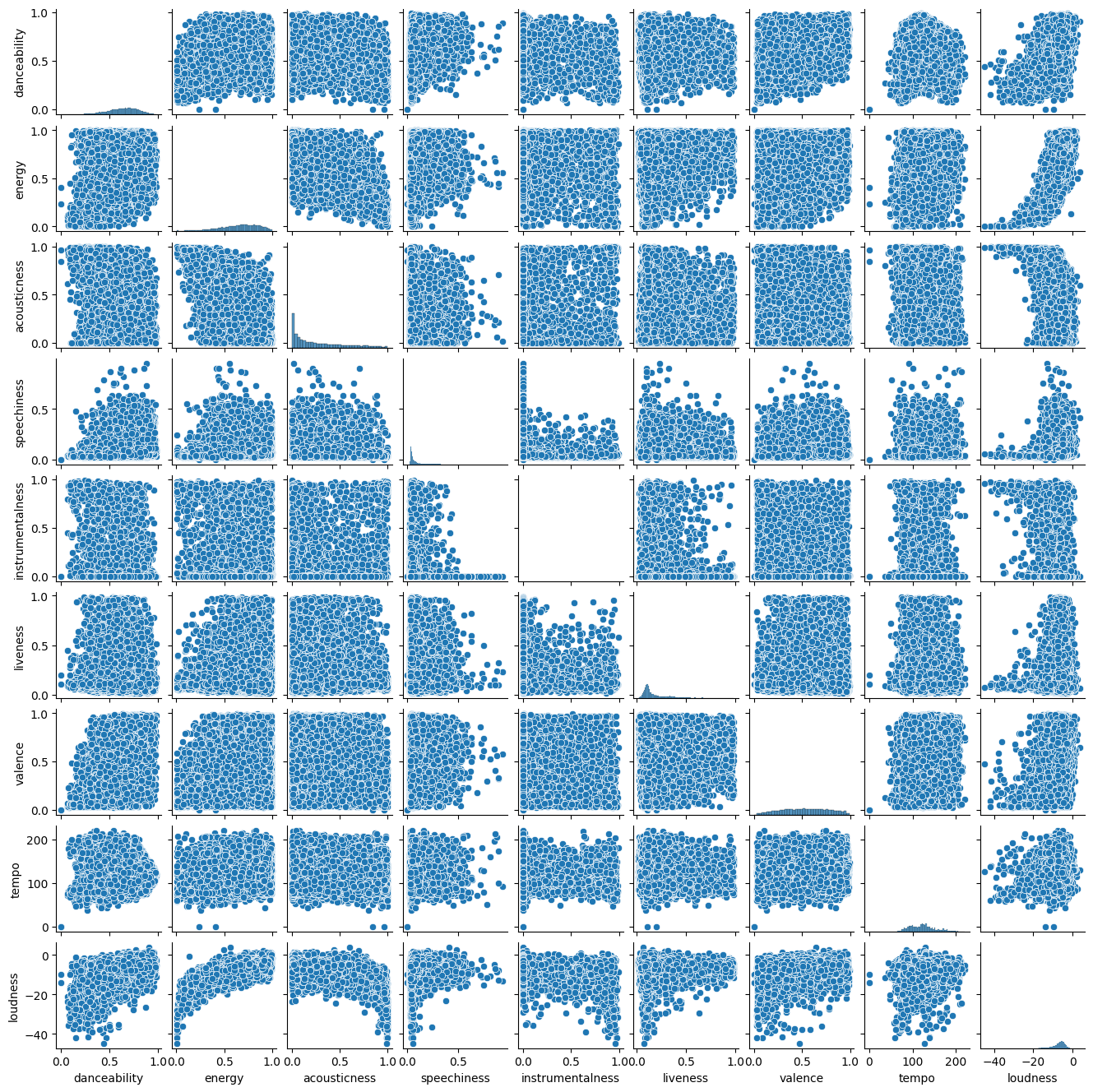
[7] Spotify, “80's and 90's rock-rock playlist greatest hits”, Dec. 2024. [Online]. Available: https://open.spotify.com/playlist/7wngYk7z50vFOXoaI5EIJM

**Appendix:**

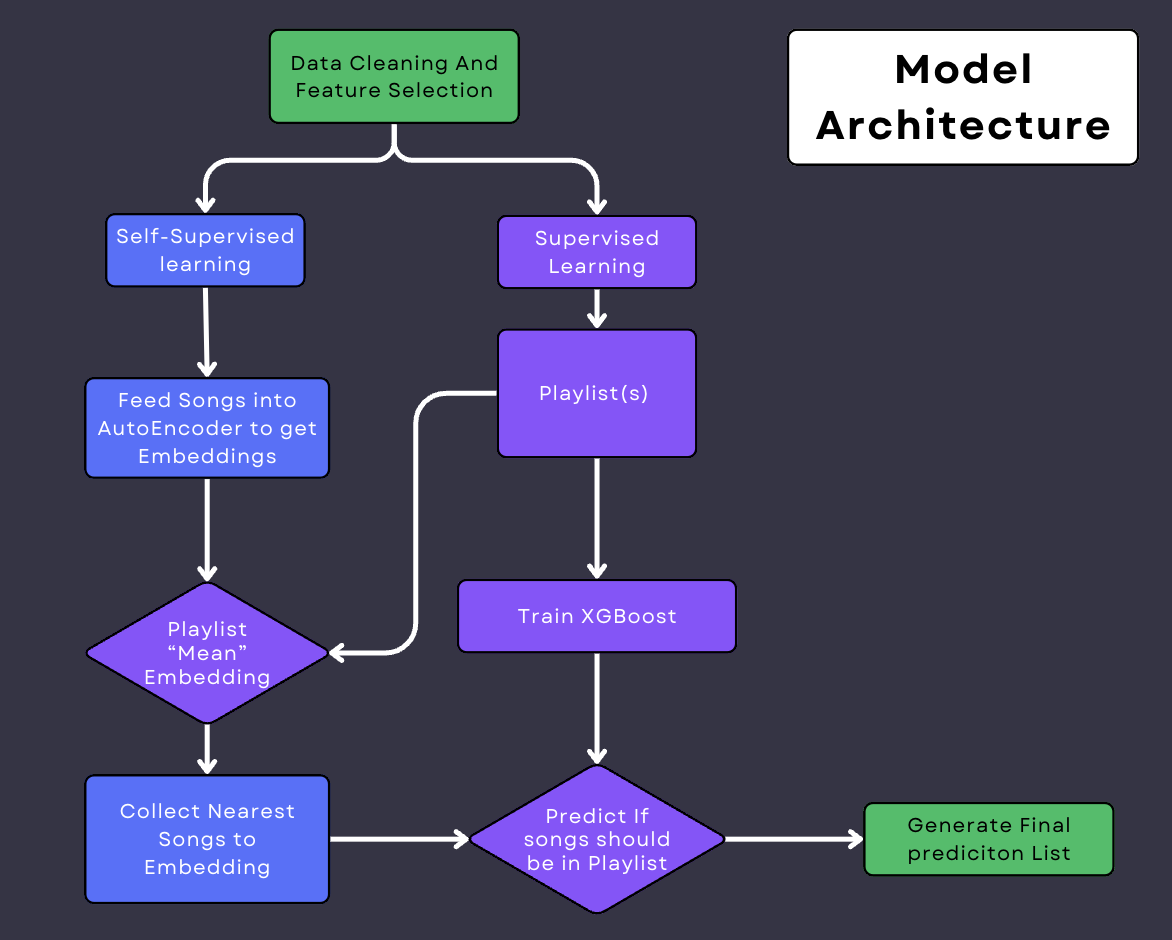
Appendix A: Correlation Matrix of Features

****

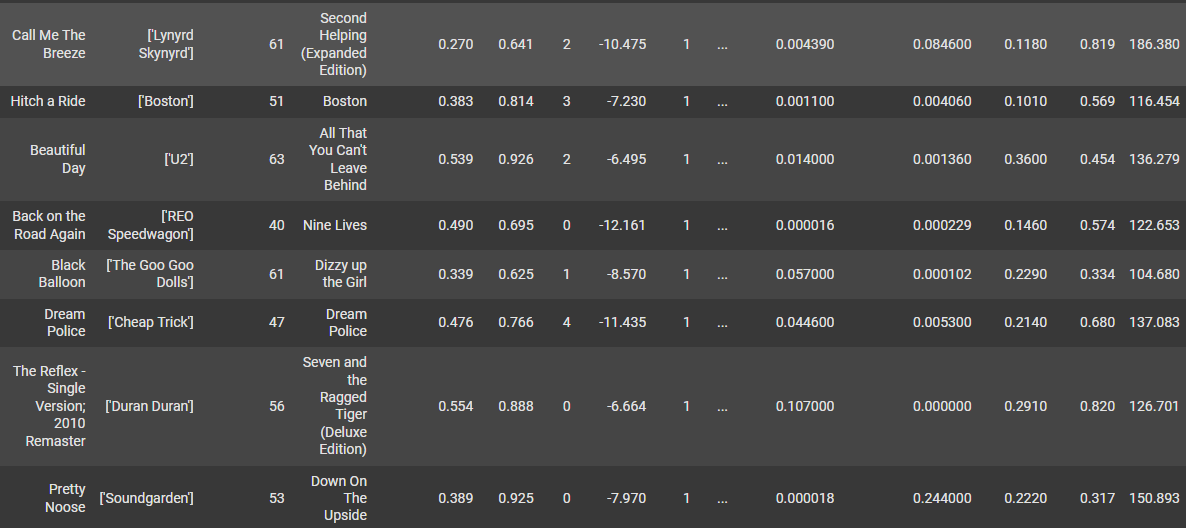
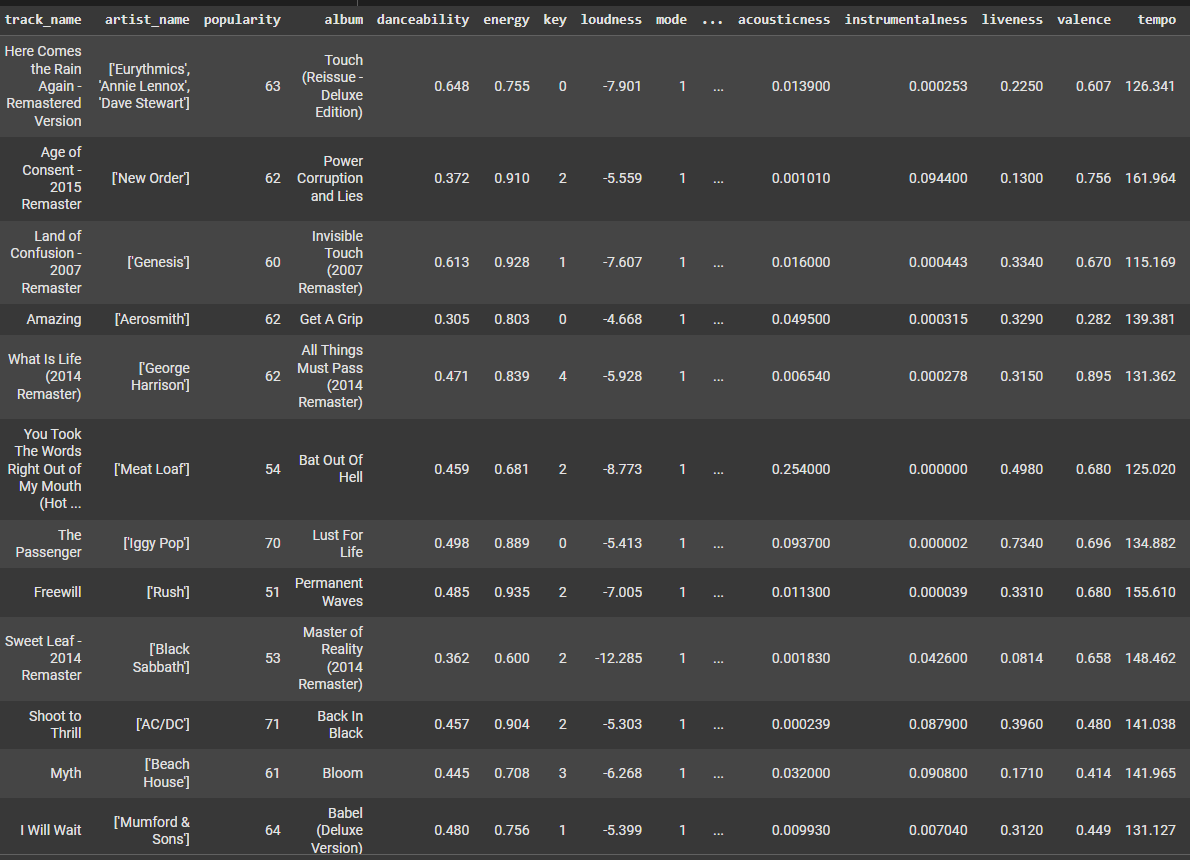
Appendix B: Pairplot of Features

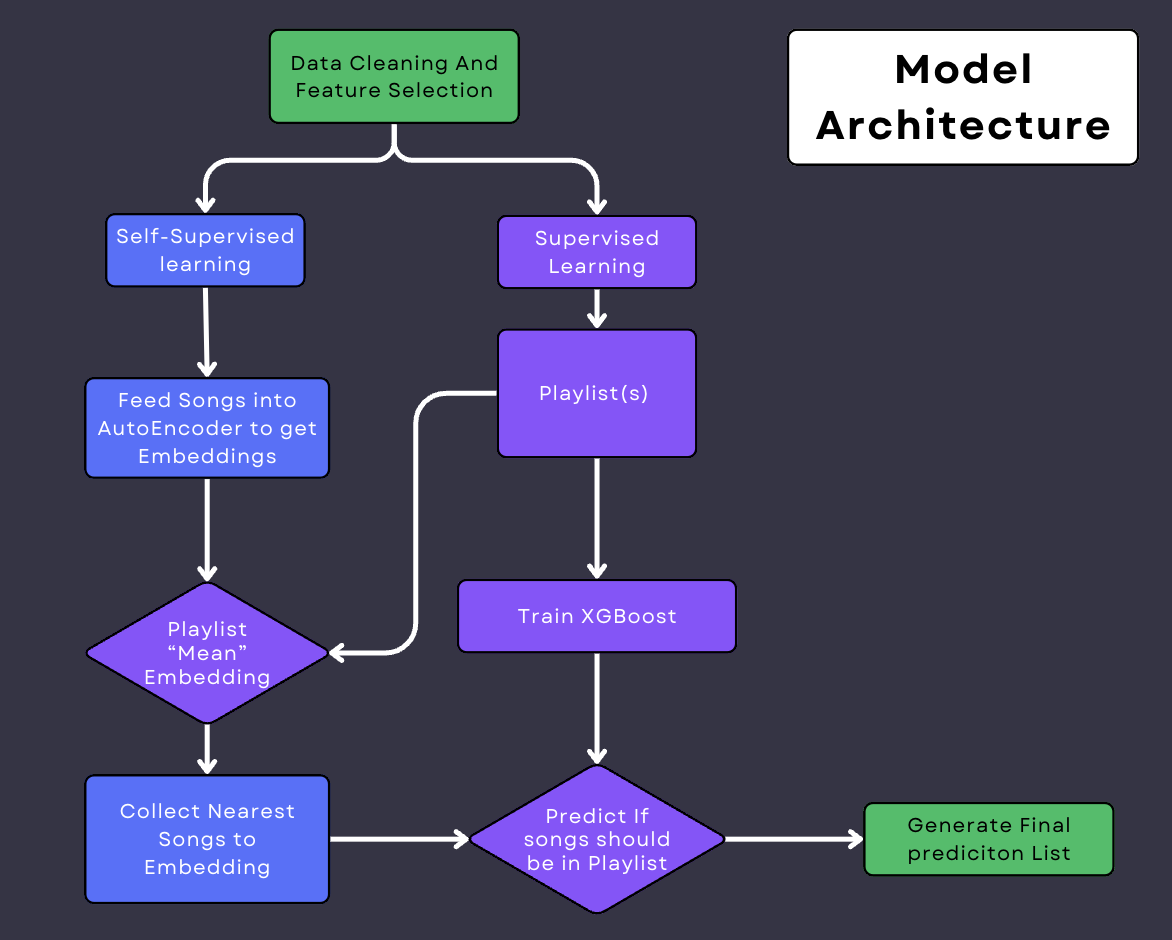
****

Appendix C: Overall model architecture, presented as a flow chart



Appendix D: 20 Recommended Songs for a Rock Playlist



Appendix A: 20 Recommended Songs for a