Mood-Dynamic Playlist: Interpolating a Path of Emotions Using a KNN Algorithm

By
Shaurya Gaur

A THESIS

submitted to

Oregon State University

Honors College

in partial fulfillment of
the requirements for the
degree of

Honors Baccalaureate of Science in Computer Science
(Honors Scholar)

Presented May 22, 2023
Commencement June 2023

AN ABSTRACT OF THE THESIS OF

Shaurya Gaur for the degree of Honors Baccalaureate of Science in Computer Science presented on May 22, 2023. Title: Mood-Dynamic Playlist: Interpolating a Path of Emotions Using a KNN Algorithm

Abstract approved:

Patrick Donnelly

Users curate music playlists based on emotion to focus or relax, so streaming services often create playlists of songs to aid this process. Prior research focuses on generating playlists of a single mood or genre, although a few studies work to construct automatic playlists that transition between the genres of their first and last songs. Building upon previous methods, we present a novel method for efficiently generating smooth and even playlists dynamic in both mood and genre. Given starting and target songs and a desired length, our two-stage algorithm sequentially chooses each track in between. It first uses a K-Nearest Neighbor model to retrieve potential songs based on emotional annotations in the Valence-Arousal emotion space, and then a vector distance metric to choose the next song based on audio features. We evaluate the smoothness and evenness of playlists across various regions of an emotional feature space. Our experiments indicate that the quality of track transitions from our algorithm's playlists is dependent on various input parameters and the distribution of the songs in our emotion dataset. This algorithm has potential as a novel music therapy tool, allowing users to create playlists in specific genres that nudge them to a desired mood.

Key Words: Music Information Retrieval, Music Emotion Recognition, Automatic Playlist Generation, K-Nearest Neighbor

Corresponding e-mail address: shauryavrat@live.com

©Copyright by Shaurya Gaur
May 22, 2023

Mood-Dynamic Playlist: Interpolating a Path of Emotions Using a KNN Algorithm

By
Shaurya Gaur

A THESIS

submitted to

Oregon State University

Honors College

in partial fulfillment of
the requirements for the
degree of

Honors Baccalaureate of Science in Computer Science
(Honors Scholar)

Presented May 22, 2023
Commencement June 2023

Honors Baccalaureate of Science in Computer Science project of Shaurya Gaur presented on May 22, 2023.

APPROVED:

Patrick Donnelly, Mentor, representing Computer Science

Robin Hess, Committee Member, representing Computer Science

Mei-Ching Lien, Committee Member, representing Psychology

Toni Doolen, Dean, Oregon State University Honors College

I understand that my project will become part of the permanent collection of Oregon State University Honors College. My signature below authorizes release of my project to any reader upon request.

Shaurya Gaur, Author

Contents

1 Introduction **3**

2 Modeling Music and Emotions **5**
2.1 Models of Emotion 5
2.2 Representing Music as Data 6

3 Distance-Based Algorithms **8**
3.1 Distance Metrics 8
3.1.1 Minkowski Distances 8
3.1.2 Cosine Similarity 9
3.1.3 Jaccard Index 9
3.2 K-Nearest Neighbors 9

4 Related Work **11**
4.1 Playlist Continuations 11
4.2 Dynamic Playlists 11
4.3 Mood Playlists 12

5 Datasets **13**
5.1 Music Emotion Datasets 13
5.2 Audio Feature Datasets 15
5.2.1 Spotify Audio Features 15
5.2.2 Million Song Dataset 18
5.2.3 Spotify Audio Segments 20

6 Preprocessing **23**
6.1 Principal Component Analysis 23
6.2 Data Cleaning Process 24
6.3 Song Dataset Class 26

7 Mood-Dynamic Playlist Algorithm **27**
7.1 Stage 1: K-Nearest Neighbors 29
7.2 Stage 2: Distance-based Song Selection 30
7.3 Time Complexity Analysis 31

8 Evaluation **32**
8.1 Experimental Design 32
8.2 Evaluation Metrics 33
8.2.1 Smoothness: Pearson Correlation Coefficient 33
8.2.2 Evenness: Step Size Variance 34

9 Experiments & Results **36**

9.1 Distance Metrics 36
9.2 Audio Datasets 38
9.3 K Values 42
9.4 Playlist Lengths 43
9.5 Dataset Segment Counts 45
9.6 Regional Variances 47

10 Discussion **49**
10.1 Summary of Findings 49
10.2 Limitations 50
10.3 Future Work 50
10.4 Applications 51
10.5 Conclusion 51

References **52**

1 Introduction

Imagine yourself feeling frustrated and tense, late into the night. You need to go to sleep soon in order to wake up early in the morning, but you have far too much energy or are in a foul mood. To help yourself relax, you might listen to music. Studies have shown that listening to music for as little as five minutes can improve a user's mood and energy level [27], revitalize focus and productivity [34, 38], and break the monotony of work [48]. Music has been shown to be effective in elementary school classrooms as well, encouraging positive and productive behavior among children [73]. Using music can help children stay relaxed [23] and even provide a means of therapy for children with autism or dyslexia [44].

Research in the psychological effects of music has found that different types of music have varying impacts on a listener's mood. Faster songs [26] with a major-key [6] are linked with happiness [23], and more energetic music such as rock can increase a listener's stress and adrenaline, which can distract from exercise pain [26]. However, these effects also depend on the listener, as background music is more likely to distract introverted people than extroverted people [28]. Additionally, a listener's familiarity with a song can foster productivity in writing [18].

Recommending music for users is an important task in the field of automatic playlist generation. Streaming services such as Spotify [33] and Deezer [8] generate automatic playlists for listeners. Most approaches in this field rely on data collected from tracks and users, including songs' audio characteristics [17], the popularity or genres of their artists [37], and user listening history [49]. Given this data, music recommendation systems often perform one of two tasks: choose the next song to play given a user's listening history, or produce a playlist of songs for them to enjoy.

Ongoing research efforts in this area strive to choose songs that are similar in features or mood to the current song being played. This requires tools and metrics that determine how similar two songs are to each other. Many approaches represent songs as a vector of relevant features, and calculate their distances to each other using metrics such as the Euclidean [22], Jaccard [2], and Cosine [42] scores. These allow researchers to use a K-Nearest Neighbor (KNN) approach such that a model returns a playlist of the K closest songs to one in the user's listening history or library [46, 49]. Some approaches use data which maps songs to emotions, which are in turn used to create playlists of tracks with a similar mood [10, 17].

While most studies focus on recommending similar songs, recent approaches also aim to create dynamic playlists which smoothly transition between songs or artists. Some of these approaches build graphs of similar artists [37, 63], which allow playlists to follow a path from one artist in one genre to another artist of a different genre. In one approach, Flexer et al. [25] use probability models to choose a playlist that

bridges the gap between two songs of different genres. However, this method only uses the genre categories of songs to evaluate the transitions between tracks in a playlist.

In this thesis, we present a novel algorithm that builds playlists which gradually transition from the mood of an origin song to that of a destination song. First, we collect and analyze public datasets of songs and their values in a two-dimensional emotional space [16] alongside their audio features [5, 69]. We then apply a K-Nearest Neighbor model and various distance scores in a two-stage algorithm, which analyzes our data to choose songs that lie in a linear path between the start and end songs of a playlist. To our knowledge, this method is the first to successfully generate playlists that are dynamic in both the mood and genre of the songs within it.

To evaluate the transitions between songs, we identify two key traits in a successful dynamic playlist: smoothness and evenness. We present a novel approach to measuring these qualities empirically by analyzing playlists as points in a continuous space. In our experiments, we generate a large number of playlists that travel between various moods to examine the effectiveness of our algorithm in various conditions. We test the impact of various factors in our playlist algorithm, including the recall of the KNN model, the distance metric to compare songs, and the dataset of audio features.

Our algorithm can allow users to create playlists that slowly nudge them from their current mood to a desired state of mind. This approach will enable development of powerful new therapeutic tools to help users gain focus to work [18], calm down before sleep [73], maintain intensity in exercise [26], or improve happiness [61].

2 Modeling Music and Emotions

Research in computational methods of emotion recognition have produced models that represent emotions as a set of categories and as a spectrum of values in a multi-dimensional space. Similarly, investigators in the field of music information retrieval (MIR) have created models that distill a song's audio characteristics into numerical features. This foundational background enables computational methods, such as our algorithm, to interface with mood and audio to generate playlists that use music to travel through emotions.

2.1 Models of Emotion

To build technology that effectively interfaces with human emotion, researchers have established various frameworks to model different moods. Studies in emotion classification often examine emotion as either a series of discrete categories, or as multi-dimensional points in a continuous space. Two of the most popular approaches in discrete categorization are Ekman's six basic emotions [20] and Plutchik's concentric circle model [57]. For continuous representations, Russell's circumplex model [62] maps emotions as points in a multi-dimensional space by their Valence (positivity) or Arousal (intensity). This representation of affect has been used in several music emotion studies [10, 17, 36, 46].

Discrete frameworks allow researchers to classify emotions into a handful of categorical labels, shown in Table 1, which can be relatively easily learned by classification models in modern machine learning (ML) algorithms. In contrast, predicting continuous emotions requires regression, which has been shown to be a more difficult task in ML. Studies that attempt to recognize emotions from facial expressions have reported an accuracy of over 80% when classifying discrete emotions [19, 65], but found far weaker performance when trying to predict continuous values in the Russell continuous space [36]. For recognizing emotions, studies have found that mapping discrete labels from Plutchik and Ekman to the Russell space [52], or using quadrants of this space [50] have also been effective in classifying emotions.

	Author	Basic Emotions
[20]	Ekman	Anger, Disgust, Fear, Happiness, Sadness, Surprise
[57]	Plutchik	Joy, Sadness, Anger, Fear, Trust, Disgust, Surprise, Anticipation

Table 1: List of the simple emotions proposed by Plutchik and Ekman.

However, using only six or eight categories fails to capture the full spectrum and granularity of human emotion. The ability to better approximate this granularity is a

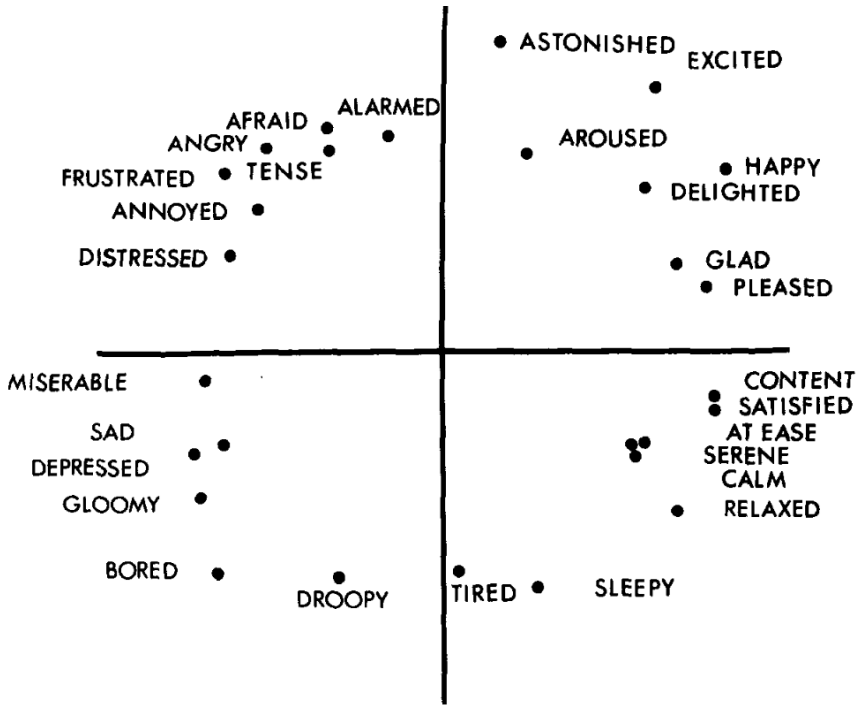


Figure 1: Russell's circumplex model of emotion (from [62]).

strength of continuous models, which have been used in recent studies on mood-based playlist generation [10, 46]. These models typically have modeled emotions in the Valence and Arousal dimensions, as seen in Figure 1, but some studies also augment these with Dominance (regarding the locus of control in an emotion) or harmonic resonance, which can factor in a user's satisfaction in a song [17]. These dimensions and models are often interrelated, as Russell claims that the emotions from Ekman's and Plutchik's models can be represented in the Valence-Arousal plane [62].

2.2 Representing Music as Data

Music can be represented in a variety of ways, and studies in MIR focus on extracting the most essential features of songs from raw audio, symbolic music scores or user-based data. Many studies use data extracted from a song's raw audio waveform [30], such as the tempo (often represented in beats per minute or BPM), pitch (commonly represented through 12 classes) [31, 30, 22], and Mel-Frequency Cepstral Coefficients (MFCC) [70, 72] to represent timbre. This numerical data is popularly found in large datasets such as the Million Song Dataset [5] or through the application programming interfaces (API) of streaming services such as Spotify [69]. Other studies [67, 22] use MIDI data to represent the musical notes of songs, though MIDI-based datasets in MIR have often been limited to genres such as folk or jazz.

While many studies use data that represents the audio features of songs, some collect user-created descriptions from social media platforms like Twitter [55] or album review scores from Pitchfork [56]. Other approaches collect descriptive tags from music information services like Last.FM [16], DBPedia or MusicBrainz [2]. Data is often collected at the artist [2, 37], album [56] or playlist [42] level as well. While these datasets are descriptive and plentiful, a 2020 study by Daikoku [15] found that current MIR methods do not generalize well to non-Western music, underscoring the lack of cultural and stylistic diversity in these widely-used music datasets.

Representing the audio and emotional characteristics of songs numerically enables the use of mathematical formula to calculate how similar or different songs are to each other. These methods compare songs as points in a multi-dimensional vector space, allowing researchers to represent playlists as a collection of points.

3 Distance-Based Algorithms

A key task in automatic playlist generation is determining how similar one song is to another. To model this similarity, researchers employ mathematical scores that determine the distances between songs, and store this information in models that allow developers to efficiently retrieve the closest songs to a target.

3.1 Distance Metrics

Techniques in MIR rely on mathematical metrics that calculate the differences between songs, based on their data from a variety of features. Several studies calculate the discrete differences of MIDI notes [67, 22] or pitch values [30, 9] through different song segments to identify transformations within a song or between two songs. Others [31] create Gaussian probability models to predict discrete genre values, representing songs as distributions to compare using the Kullback-Leibler (KL) divergence. Some studies build graphs [45, 41, 37, 63] to model similarity, such as the MusicLynx[1] platform [2], a visualizer that connects similar artists based on a weighted metric that compares their genres.

While researchers employ various methods to compare discrete or categorical data, our work focuses on a song's numerical features. These represent songs as vectors in a feature space, and allow us to use mathematical metrics to calculate the distance between them. Some scores calculate vector similarity instead of distance, which we easily convert to distance metrics by subtracting these similarities from their maximum values. We examine three metrics below.

3.1.1 Minkowski Distances

Perhaps the most popular vector distance metrics in mathematics are the Manhattan and Euclidean distances [22, 72, 46]. These are specific versions of the Minkowski distance [68], a formula for calculating the distance between two vectors in a space of a specific order, as shown in Equation (1).

$$D(X,Y) = (\sum_{i=1}^{n} |x_i - y_i|^p)^{\frac{1}{p}} \tag{1}$$

This is equivalent to the p-norm of a vector $X - Y$. The Manhattan distance, also known as the taxicab distance, uses $p = 1$, which is the sum of the distances between two vectors in every single dimension. The Euclidean distance uses $p = 2$, and it is often used as the default norm for vector problems.

[1] https://musiclynx.github.io/

3.1.2 Cosine Similarity

Several researchers in playlist generation also measure the Cosine Similarity, or angular similarity, between two vectors [17, 63, 42]. This score calculates the cosine of the angle θ between two vectors. The domain of $\cos\theta$ is bound in the range of $[-1, 1]$, where a value of 1 indicates $\theta = 0$, or that the two vectors lie on the same line. Therefore, to determine the similarity of two vectors [58], we can rearrange the formula for the vector dot product to Equation (2).

$$S_C(X, Y) = \cos\theta = \frac{X * Y}{||X||\ ||Y||} = \frac{\sum_{i=1}^{n} X_i Y_i}{\sqrt{\sum_{i=1}^{n} X_i^2}\sqrt{\sum_{i=1}^{n} Y_i^2}} \tag{2}$$

To utilize this as a distance metric, we calculate $D_{SC}(X, Y) - 1 - S_C(X, Y)$ as the Cosine distance between songs in our algorithm.

3.1.3 Jaccard Index

This metric was designed to examine the similarity of two sample sets by comparing the size of their intersection (the common elements in both sets) to that of their union (the combined set of both samples) [32]. This "intersection-over-union" approach functions similarly to a Venn diagram and has been used in models that detect objects like stop signs in images [71]. MIR researchers have previously used the Jaccard index to calculate the similarities between users [55] and artists [2]. For vectors, the Jaccard index is calculated by comparing the minimum and maximum values for two vectors at each dimension, as seen in Equation (3).

$$J_W(x, y) = \frac{\sum_i \min(x_i, y_i)}{\sum_i \max(x_i, y_i)} \tag{3}$$

A value of 1.0 means that the two vectors are the same, and a smaller value indicates a greater distance between them. We calculate $D_{JW}(X, Y) = 1 - J_W(x, y)$ as the Jaccard distance in our algorithm.

3.2 K-Nearest Neighbors

In MIR studies, these distance metrics are often employed as part of larger algorithms that model the relationships between different songs or artists. One popular algorithm is the K-Nearest Neighbors (KNN) model, which makes decisions on a sample based on the values of the K nearest points around it, as determined by a distance metric. Proposed in 1951 [24], KNN was originally designed for classification problems in machine learning in which a classification for a sample point is the most frequent label among its neighbors (see Figure 2). However, this example-based learning algorithm

also found success in regression problems, where a point's numerical value is calculated as the mean of the values of its K nearest neighbors [3]. KNN-based regression have proven useful in handling non-linear patterns in data.

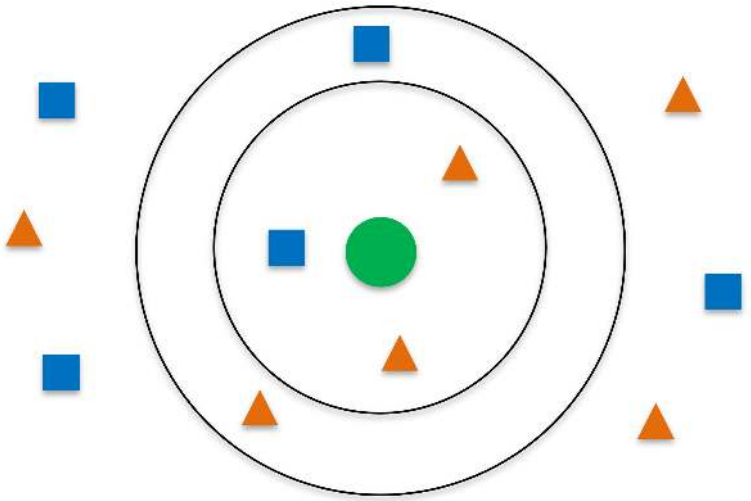


Figure 2: Example of a KNN classification problem. The green sample point is labeled as orange when $k = 3$ but as blue when $k = 5$.

Several researchers employ KNN for not only classification [51, 65] and regression [55, 47], but also to search for the neighbors of a sample or target song [46]. We employ this latter approach and utilize an unsupervised `NearestNeighbors` model[2] provided by the `scikit-learn` Python package [54], a popular collection of tools used in machine learning. This model uses efficient data structures to store neighboring songs, varying its approach based on the number of songs and the dimensionality of their features. When a dataset has few points, or the distances between points are already computed, this model stores the distances between every pair of points. However, this approach becomes unfeasible with a large dataset, so the `scikit-learn` team resorts to tree-based approaches, which group nearby points to each other [54]. Often, these are K-dimensional trees, which partition each dimension in half based on its median value. This effectively creates quadrants for two-dimensional data, or octants for 3-dimensional data [4]. Constructing this tree and returning neighbors is fast for fewer dimensions but grows inefficient with higher-dimensional data. In these cases, the `NearestNeighbors` model uses a Ball Tree, which clusters data into spheres of points that extend from various centroids [54]. The adaptability and efficiency of KNN models make them an effective tool in automatic playlist generation.

[2] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.NearestNeighbors

4 Related Work

People have curated lists of songs to play ever since we started listening to music. As a result, the MIR community is highly motivated to develop methods that automatically create playlists [12, 17, 10, 25] or recommend the next song for a user to play [49, 42, 46, 75]. These researchers build models based on data from songs, or in the case of streaming platforms like Spotify, data from users' listening tastes [46] or existing playlists [42, 75]. Some approaches also create playlists of emotionally similar songs [17, 10, 45], or playlists that gradually transition in audio features from an origin to a destination song [25, 63, 12].

4.1 Playlist Continuations

Playlist generation can be seen as a continuation problem, where a model recommends the next song to play given a list of the user's recently played tracks. Many approaches exist for these problems, such as Pampalk's method [49] of comparing the similarity of songs' audio features to previously played and skipped songs to choose the next song for the user.

Contestants in the 2018 ACM Recommendation Systems challenge used various methods to choose the next track, given a playlist of previous songs from a Million Playlist Dataset (MPD) provided by Spotify [42]. Many techniques used collaborative filtering (CF) to choose tracks based on their occurrence in other similar playlists [42]. CF can be a popular approach in playlist generation, and has been employed by Spotify [33] and Pichl et al. [55] to recommend songs based on the listening histories of other users. While this approach has proven effective with large datasets and popular streaming services, it suffers from a cold-start problem when it lacks initial users and skews towards popular tracks [46].

In contrast to CF, Ludewig [42] found that an approach using matrix factorization algorithms and KNN techniques was simpler, yet almost as effective, as more complex natural language processing (NLP) or convolutional neural network (CNN) systems. Zamani's examination of approaches in this challenge [75] found that the highest performing systems created a two-stage algorithm to recommend songs. This type of system uses a high-recall first stage to choose many candidate songs, and then re-scores these in a second stage, in which the most relevant candidate is chosen as the next song.

4.2 Dynamic Playlists

Previous studies have employed various methods to create playlists that smoothly transition between the audio characteristics of start and end songs. Pauws [53] and Chen [12] gather data on existing playlists using Markov embeddings or simulated an-

nealing to build new playlists that optimize song transitions. Sakurai [63] constructs similarity graphs based on songs' audio features using Reinforcement Learning to create playlists with high diversity and smooth track transitions. On the other hand, Lemare [37] creates playlists that transition between artists, using data on artist popularity and the energy of their tracks to determine a playlist of songs.

Flexer [25] creates playlists that gradually transition from a start to an end song, modeling songs as Gaussian probability models based on MFCC features. Their approach computes each song's KL divergence from the start and end song to create a playlist of songs in between. However, they only evaluate the transition between the playlists' first, middle, and last three songs, and only examine transitions in discrete genre categories rather than numerical audio features.

4.3 Mood Playlists

Various studies have worked to classify the emotions of users and songs to recommend music of similar emotions. Rumiantcev [61] designs a high-level model of a system which uses user and sensor data to recommend songs similar to a user's predicted mood using KNN or Random Forest approaches. Several other studies take strides in implementing such a design. Meyers [45] models the moods of artists, albums and songs into several discrete categories using a graph-based method [41] and compares these emotions to those extracted from mapping the words from a user's writing. EMOSIC [47] is a music player that performs similar emotion recognition on a user's face and songs' audio. They use Support Vector Machines and KNN approaches to predict values in the Valence-Arousal plane, and recommend songs emotionally similar to each other and the user's current mood.

While the above studies use data on songs, users, and albums, some other studies employ a purely song-based approach. Deng [17] predicts the emotions of the user's previously-listened songs from audio features to continue their current playlist, while Pao [51] implements a weighted-density KNN model to map emotions into discrete quadrants of the Valence-Arousal space. Moscato's approach [46] uses data from the Deezer 2018 dataset [16] and MFCCs to train a CNN to predict the Valence-Arousal mood values of songs outside the dataset. Their system uses a KNN model powered by the Euclidean distance to continue playlists with emotionally similar songs. While most studies recommend music to users that maintains their current mood, Cardoso [10] enables users to draw a path in the Valence-Arousal plane. Their approach chooses the closest songs to that path within a certain threshold, yet does not explicitly order these songs or evaluate the quality of these path-based playlists.

	All Songs (N=18,644)		**Unique Points** (N=2,720)		
	Valence	*Arousal*	*Valence*	*Arousal*	*Frequency*
Mean	-0.067258	0.195720	-0.370665	0.059615	6.854412
St.Dev.	1.057872	0.960662	0.798128	0.722372	59.394308
Min	-2.148097	-2.333604	-2.148097	-2.333604	1
Median	0.032224	0.040198	-0.464615	-0.014126	1
Max	1.546714	2.755091	1.546714	2.755091	1732

Table 2: Summary statistics for all Deezer songs (left) and unique points (right).

5 Datasets

To create playlists that are dynamic in emotion and audio, we require datasets which contain these features for a large number of songs. These datasets will function as a space in which our algorithm will select songs to form a path, representing a playlist. We combine datasets of songs and their points in both the Valence-Arousal emotional space, and various audio-based feature spaces.

5.1 Music Emotion Datasets

Many different datasets have been used for MIR and Music Emotion Recognition (MER) which provide ratings for songs in a continuous emotional space. To generate playlists that travel through emotions, we seek datasets that include information on songs and their values in the Valence-Arousal space. Typically, high-quality MER datasets conist of manual ratings provided by human annotators listening to music [11, 76, 1], but these only provide such data on either a few hundred songs. To find a sufficiently large dataset for our task, we primarily use the Deezer 2018 [16] dataset of 18,644 songs with synthetic values for valence and arousal.

Researchers from the Deezer music streaming service collected mood-related tags from the Last.FM service to describe each song, and used a dataset to map these words to their embeddings in the Valence-Arousal emotion space. They collected and normalized the means of these embeddings, releasing valence and arousal labels, IDs for the songs on Deezer and the Million Song Dataset [5], and the artist and track names for all 18,644 songs.

However, the synthetic nature of this dataset's ratings means that songs with the same or similar Last.FM tags may share the same value in the Valence-Arousal space [16]. This is a major limitation of this dataset, as the 18,644 songs only occupy 2,720 unique points. Table 2 shows that around seven songs on average share an emotional point, with as many as 1,700 songs occupying a single point.

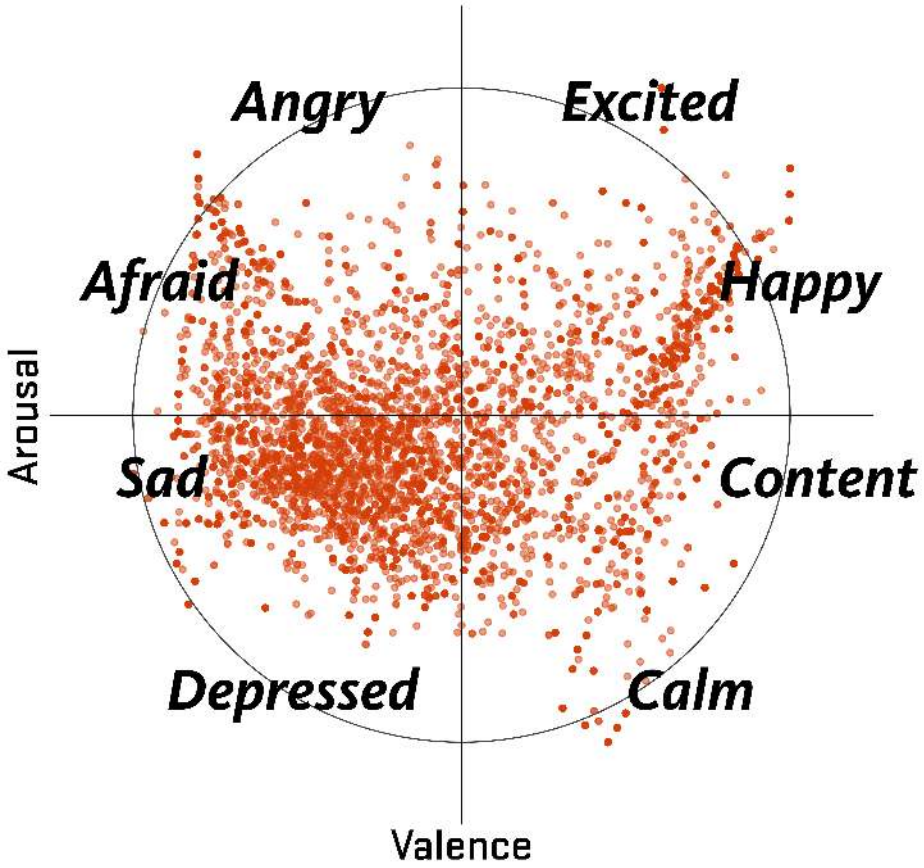


Figure 3: Scatterplot of the distribution of Valence and Arousal in the Valence-Arousal circumplex space, each scaled to the [-1,1] range.

Additionally, Table 2 and Figure 3 show that the distributions of the points in the dataset are uneven. While the distribution of Valence scores seems standardized between songs, this changes when filtering only for the unique points, which shows that happy and upbeat songs are more likely to share the same point in the Valence-Arousal space. The opposite is true for the Arousal values, which show a skew towards high-energy songs, but are fairly standardized when only considering unique points.

These two skews compound each other when considering the quadrants of the dataset. Table 3 shows that the Deezer dataset is more skewed towards songs that are either positive or negative in both valence and arousal, but sad and depressing songs have far more unique points in this space than any other type of song. This creates various regions in our dataset that are denser and sparser. Despite these limitations, Deezer's set of popular songs provides a large sample from which to build playlists and retrieve audio features from other public datasets.

Quadrant	Valence	Arousal	Emotions	Songs	Points
I	+	+	Excited, Happy	6001	502
II	+	-	Angry, Afraid	3638	832
III	-	-	Sad, Depressed	5444	1047
IV	-	+	Calm, Content	3561	339

Table 3: Descriptions and the number of songs and unique points in each Cartesian quadrant of the Deezer 2018 dataset.

5.2 Audio Feature Datasets

Since user preferences in genres of music have an effect on their mood, our goal is to create mood-dynamic musical playlists that also traverse smoothly between the genres of chosen origin and destination songs. Therefore, we augment our music emotion data of the songs from the Deezer dataset with acoustic features and other relevant metadata from the Spotify API [69] and the Million Song Dataset [5]. We test these feature spaces in Section 9.2.

5.2.1 Spotify Audio Features

As a first step, we extract additional features from Spotify, a large, popular streaming service for songs, with more than 500 million users, including 205 million subscribers in more than 180 markets[3]. The company uses collaborative filtering algorithms augmented with audio feature data driven by its 2014 acquisition of the Echo Nest API, which specialized in extracting feature embeddings from a song's audio and metadata [50]. The Spotify API includes endpoints for fetching audio features and other metadata for almost every song in its catalog [69].

To match a song from the Deezer dataset to a track from Spotify's catalog, we queried the API for a Spotify track ID using a search term that concatenates the artist and title of each song. This successfully queried for the URI for 17,755 of the Deezer dataset's 18,644 songs, and 2,672 unique points out of a total of 2,720. All audio-related features from the Get Audio Features[4] API endpoint were extracted. From the Get Track[5] API endpoint, we extract the popularity metric and the explicit tag of a track, since the popularity or explicit content of a song may have an effect on the user listening experience.

[3] newsroom.spotify.com/company-info/

[4] developer.spotify.com/documentation/web-api/reference/get-audio-features

[5] developer.spotify.com/documentation/web-api/reference/get-track

	Mean	St.Dev.	Min	Median	Max
acousticness	0.2618	0.3051	0	0.1110	0.995
danceability	0.5242	0.1590	0	0.5270	0.978
duration_ms	251336	81137	35587	239	1449973
energy	0.6222	0.2462	0	0.6510	0.999
instrumentalness	0.0844	0.2089	0	0.0002	0.983
key	5.2506	3.5622	0	5	11
liveness	0.1878	0.1557	0.0136	0.1250	1
loudness	-8.4547	3.9755	-46.2840	-7.6410	3.744
mode	0.6823	0.4656	0	1	1
speechiness	0.0605	0.0632	0	0.0387	0.954
tempo	122.0198	29.2322	0	120.1160	217.520
time_sig	3.9018	0.3680	0	4	5
valence	0.4609	0.2481	0	0.4360	0.984
popularity	34.1625	16.6780	0	33	84
explicit	0.0444	0.2059	0	0	1

Table 4: Statistics for 13 features from Spotify the Get Audio Features endpoint, and two features from the Get Track endpoint, on 17,755 Deezer songs collected.

Overall, we extracted 15 features from Spotify, listed in Table 4. From this, we can see that most features have relatively little skew, except for the duration of a song in milliseconds. Additionally, most features are floating-point values between 0.0 and 1.0, representing confidence values for the feature described in the endpoint. However, some features represent different values at different scales:

- **duration**: *integer* value representing the song's duration in milliseconds.
- **key**: *integer* value from -1 to 11, representing the key of a track using Pitch Class Notation, with no key detected represented as -1.
- **loudness**: *floating-point* value, representing the average decibel level of the track, ranging from -60dB to 0dB.
- **mode**: *binary* value, where a major-mode track is represented as 1, and a minor-mode track is a 0.
- **tempo**: *floating-point* value representing the estimated tempo in beats per minute (BPM).
- **time_sig**: *integer* value between 3 and 7, representing a 3/4 to 7/4 time signature range.

- **popularity**: *integer* value from 0 to 100, estimating a song's popularity based on the number of times the song was recently played at the time of request.
- **explicit**: *boolean* value, 1 if a song is tagged as explicit, 0 otherwise.

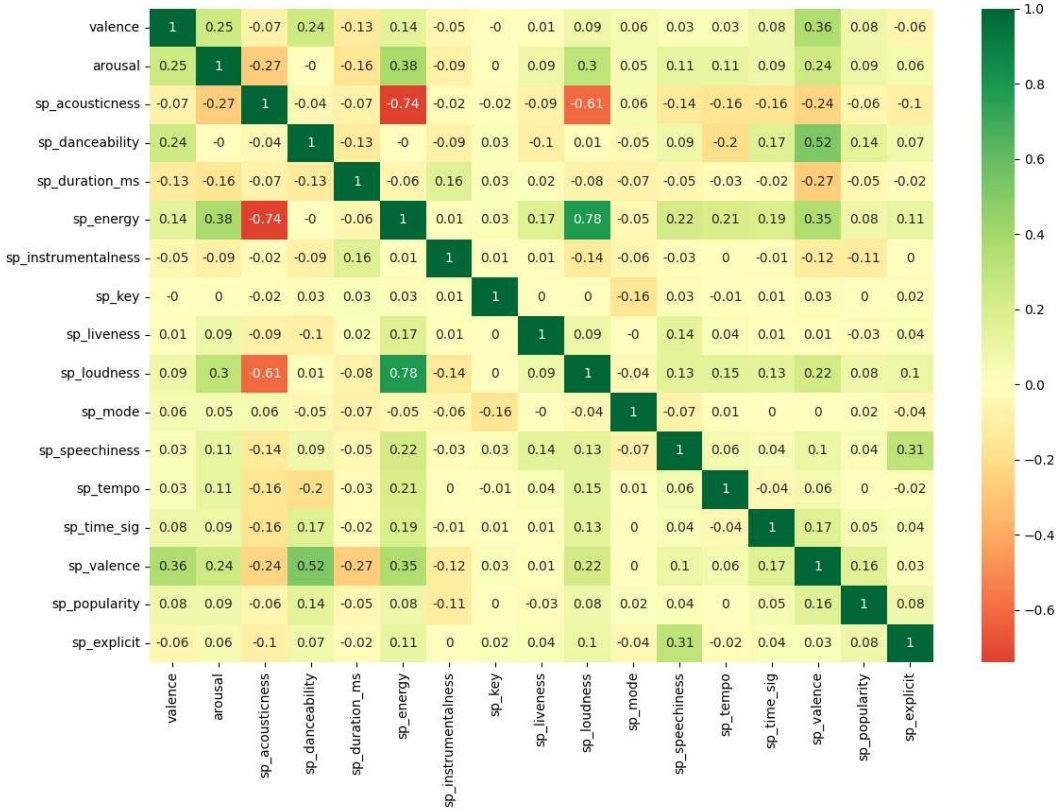


Figure 4: Correlation matrix for Valence and Arousal from Deezer 2018 and all collected Spotify features (prefixed with `sp_`).

The correlation matrix in Figure 4 shows that these features are mostly uncorrelated, save for energy, loudness, and acousticness. A 2021 study from Panda et al. found that these three features are the most relevant to MER [50]. Their study used an audio dataset of 900 songs to compare the effectiveness of Spotify features to other raw audio features in MER. This was measured by the correlation between each feature and synthetic labels for the Russell space quadrants which were validated by volunteers [50]. While the acousticness, valence, energy, and mode features had relatively high correlations individually, the whole set of 12 Spotify features had a lower correlation with labeled emotions than other feature datasets, and were thus subpar for predicting a song's mood [50]. While this may be a detriment to other music emotion applications, it implies that these features account for other characteristics

besides audio. This is useful in our application, since our audio feature datasets are meant to balance the genres of our playlist, and are not meant to signify emotion.

5.2.2 Million Song Dataset

One dataset frequently used in MIR is the Million Song Dataset [2, 39, 16], also known as the MSD. Created by a 2011 collaboration between researchers at Columbia University and The Echo Nest, with support from the National Science Foundation, this dataset includes audio-derived features and metadata for 1,000,000 songs from 44,745 unique artists [5]. Using the included `MSD_track_id` and `MSD_sng_id` for each track on the Deezer 2018 dataset, we extracted primarily audio-based fields from the MSD[6].

For each track overall, we extracted the following features from the MSD:

- **key**: *integer*, predicted key of the song.
- **mode**: *integer*, major or minor key of the song.
- **time_signature**: *integer*, estimate of the number of beats per bar.
- **key_confidence**: *floating-point*, confidence measure of the key estimation.
- **mode_confidence**: *floating-point*, confidence measure of the mode estimation.
- **time_signature_confidence**: *floating-point*, confidence measure of the time_signature estimation.
- **loudness**: *floating-point* overall loudness in decibels.
- **tempo**: *floating-point* estimated tempo in BPM.

In addition, we aggregated the loudness and timbre features of all "segments" of our song. A segment in a song is the smallest unit of time collected by the MSD, and contains a consistent sound throughout its duration [5]. We extract the loudness and timbre of segments by including the value and duration of each segment's peak decibel value, and its 12 MFCCs respectively. For each song, we calculated the average, variance, minimum, maximum, and median values across segments as a representation of an aggregated measurement of timbre and loudness. Table 5 shows descriptive statistics for all features and selected timbre aggregations[7]. While many features are relatively centered, they exist on widely different scales and could result in certain features, such as the maximum loudness and timbre, overpowering others.

[6] http://millionsongdataset.com/pages/field-list/

[7] Aggregations for MFCCs 1 through 10 not shown for brevity. Features for `loudness_max` and `timbre` are for segments and have `segments_` as a prefix in usage.

	Mean	St.Dev	Min	Median	Max
key	5.2544	3.5671	0	5	11
key_confidence	0.4727	0.2694	0	0.492	1
loudness	-8.7548	4.0393	-34.022	-7.881	-0.227
mode	0.6759	0.468	0	1	1
mode_confidence	0.5082	0.1868	0	0.519	1
tempo	122.9944	32.0993	0	120.752	262.412
time_signature	3.722	1.0331	0	4	7
time_signature_confidence	0.5543	0.358	0	0.606	1
loudness_max_avg	-12.1558	4.9524	-42.4203	-11.3151	-1.4755
loudness_max_var	36.9183	22.6442	0.5162	32.7954	270.1014
loudness_max_min	-54.0999	10.5726	-60	-60	-5.056
loudness_max_max	-3.7329	2.9353	-25.451	-3.115	6.37
loudness_max_med	-10.9841	5.0924	-48.624	-9.99	-0.864
loudness_max_time_avg	0.0607	0.0166	0.0215	0.0581	0.418
loudness_max_time_var	0.0092	0.382	0.0001	0.0032	50.7472
loudness_max_time_min	0.0037	0.0041	0	0	0.0319
loudness_max_time_max	0.8673	2.2284	0.1052	0.5913	269.3235
loudness_max_time_med	0.0432	0.01	0.019	0.042	0.1376
timbre_0_avg	44.2645	5.1914	15.2725	44.9773	55.7646
timbre_0_var	37.421	22.0484	1.1725	33.2962	249.3499
timbre_0_min	3.7664	7.7711	0	0	47.743
timbre_0_max	52.9363	3.2002	29.65	53.604	61.504
timbre_0_med	45.469	5.3997	9.568	46.308	56.13
...	...	...	...	...	...
timbre_11_avg	2.3729	7.3529	-39.9886	2.1727	47.7089
timbre_11_var	287.0263	117.2919	50.5546	267.3146	1934.4954
timbre_11_min	-56.8164	16.3461	-157.703	-55.137	-1.489
timbre_11_max	60.2111	16.8135	10.323	58.719	171.616
timbre_11_med	2.5286	7.3425	-42.053	2.309	49.9985

Table 5: Statistics for some of the 78 MSD features on 17,755 Deezer songs collected.

We extracted 78 features from MSD for all 17,755 songs from the Deezer dataset which also contained Spotify features. Because the Deezer dataset is a subset of MSD, we successfully collected features for all songs queried. While the MSD is a large dataset with many features, it lacks album and song-level metadata and tags, and contains almost no world, ethnic or classical music [5]. Also, the MSD has not been updated since its release in 2011, so musical sample is restricted to tracks released between 1922 and 2011, diminishing its relevance towards future MIR studies that consider more contemporaneous music.

5.2.3 Spotify Audio Segments

The MSD and Spotify features aggregate data at the level of an entire song, which implies that a song's sound is the same throughout its duration. Yet this does not reflect the reality of music, which often starts in one way and transitions between characteristics over the course of a song. Since a playlist involves transitions between the end of one song and the start of the next, collecting audio features for the start and end of each song could potentially yield smoother transitions.

The Spotify API's "Get Audio Analysis" endpoint[8] contains data for a song broken down into various subdivisions:

- **Sections**: defined by large variations in rhythm or timbre, e.g. chorus, verse, bridge, guitar solo, etc.
  - Contains descriptions of tempo, key, mode, time_signature, and loudness.
- **Bars** and **Beats**: time intervals for bars and beats respectively.
  - Contains the start, duration, and estimation confidence of bars and beats.
- **Tatums**: represents the lowest regular pulse train that a listener intuitively infers from the timing of perceived musical events (segments).
  - Contains start, duration and confidence of estimates.
- **Segments**: defined by a consistent sound throughout its duration.
  - Contains data on peak loudness, and vectors on pitch and timbre.

We choose to extract data from segments since they are one of the smallest units of data in a song, and contain the most data, in comparison to tatums, bars, beats and sections. This is verified visually in Figure 5, generated from a community tool[9].

We successfully extracted data on segments for 17,751 out of 17,755 songs. The remaining four songs lacked lacked audio analysis data in the Spotify API. We grab the following features from the segments of each track:

- **Loudness**: peak and onset loudness values (in dB).
- **Pitch**: 12-class chroma vector, normalized between 0 and 1.
- **Timbre**: coefficients of 12 MFC basis vectors, describing song "texture".

[8] developer.spotify.com/documentation/web-api/reference/get-audio-analysis
[9] spotify-audio-analysis.glitch.me

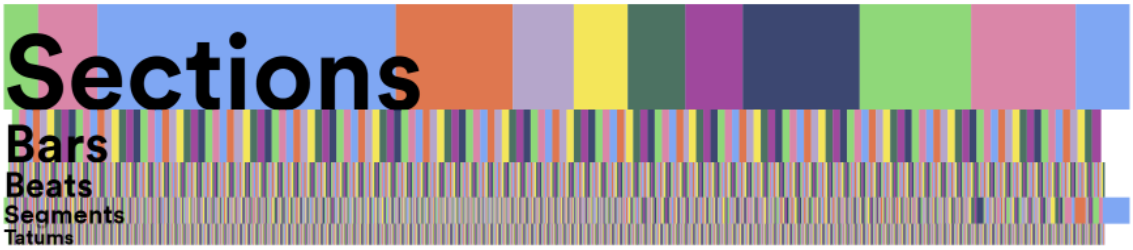


Figure 5: Visualization of subsections of *Digital Love* by Daft Punk.

For each of these features, splitting the pitch and timbre vectors into 12 distinct features each, we compute weighted averages for the head (or first N segments) and tail (or last N segments) of each song. Segments have a specific order in a song, and each segment has a varying duration in terms of the length of the song. Therefore, we decided to create a custom weighted average that considers the order − defined as the closeness to the head or tail − and duration of each segment when creating our weighted average:

$$W = \frac{\sum_{i=1}^{N-1} f_i d_i (N-i)}{\sum_{i=1}^{N-1} d_i (N-i)} \tag{4}$$

where N = the number of segments in the calculation, f_i = value of feature at the i^{th} segment, and d_i = duration of the i^{th} segment. In this formula, segments closer to the start or end with longer duration hold greater weight than later, shorter segments, since they are heard prominently at the beginning or end of a song. As more segments are factored into the weighted average, our custom weighted average value tends to stabilize around certain values. This is illustrated in Figure 6.

We collect various segment feature datasets based on the number and total duration of segments (e.g., the first 30 seconds) and round each to six decimal places. These datasets contain 26 features for the head of the song (12 from pitch vectors, 12 from timbre MFCCs, and 2 from loudness) and for the tail, for a total of 52 features. Table 6 describes the distribution of the averaged features for the first 100 segments of Deezer songs. These statistics imply that segment-based features are not skewed, but that the timbre and loudness features are at much larger scales than the pitch.

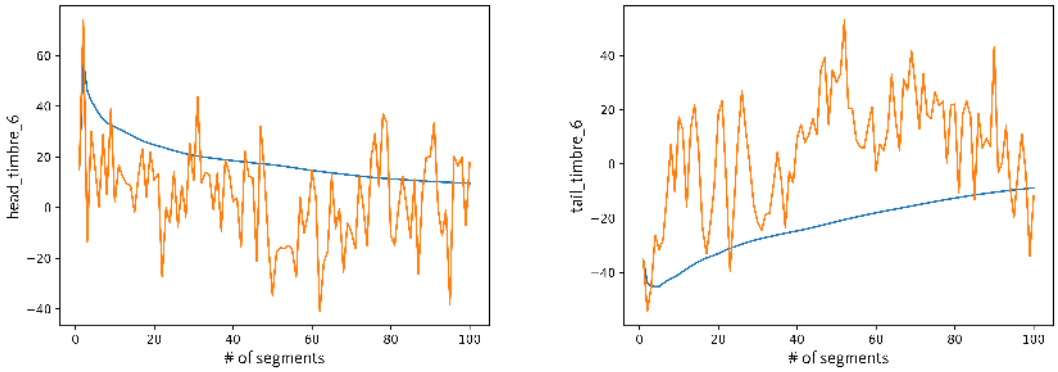


Figure 6: Custom weighted averages of up to the first and last $N = 100$ segments respectively for the `timbre_6` feature. Example song is *Gone* by Pearl Jam (id: 5JSfrkLVMTNcr4qC3jPTup).

	Mean	**St.Dev.**	**Min**	**Median**	**Max**
head_loudness_max	-16.1074	7.4773	-55.929	-15.3226	0.5292
head_loudness_start	-26.1861	7.9473	-58.8554	-26.0611	-4.4365
head_pitches_0	0.3627	0.1857	0.0098	0.3455	0.9999
head_pitches_1	0.3473	0.1834	0.0199	0.3263	0.9987
head_pitches_2	0.3264	0.171	0.0155	0.3063	0.992
head_pitches_3	0.2657	0.1485	0.012	0.2413	0.9974
head_pitches_4	0.3219	0.1681	0.0097	0.3017	0.9986
head_pitches_5	0.2842	0.1534	0.0099	0.2636	0.9845
head_pitches_6	0.2857	0.1466	0.0111	0.2658	0.9915
head_pitches_7	0.3181	0.1632	0.0112	0.299	0.9992
head_pitches_8	0.2767	0.1489	0.0148	0.2544	0.9919
head_pitches_9	0.3198	0.1688	0.0074	0.2969	0.9998
head_pitches_10	0.2593	0.1444	0.007	0.2349	0.9959
head_pitches_11	0.3114	0.1606	0.0026	0.2923	1
head_timbre_0	39.2529	7.5379	2.32	39.7333	58.4351
head_timbre_1	18.5281	61.975	-281.1501	22.838	369.1475
head_timbre_2	-5.1866	45.059	-203.8026	-4.3076	239.6
head_timbre_3	0.5451	26.534	-105.625	-3.9182	254.2491
head_timbre_4	19.9278	23.7255	-66.6107	18.5872	191.5217
head_timbre_5	-17.6264	14.7215	-106.4868	-18.8081	71.0534
head_timbre_6	-6.7193	15.0338	-72.4662	-6.8596	76.7225
head_timbre_7	-1.8159	10.8272	-71.919	-1.7113	87.3362
head_timbre_8	-7.5361	9.6454	-71.3813	-7.3055	50.1824
head_timbre_9	0.3965	6.1872	-44.9041	0.3477	45.4983
head_timbre_10	-10.9828	7.7418	-92.756	-9.7548	23.8244
head_timbre_11	1.6325	7.6403	-48.6695	1.6404	51.5329

Table 6: Descriptive statistics for 26 features for a segment dataset averaging data for the first $N = 100$ segments for 17,751 Deezer songs.

Dataset	Original Feature Count	Variance Retained
Spotify	15	96.7%
MSD	78	68.3%
All (Spotify + MSD)	93	62.7%

Table 7: The proportion of variance kept after performing a PCA to 12 features on each audio feature dataset.

6 Preprocessing

We see from Tables 2, 4, 5, and 6 that the audio feature datasets vary greatly in their dimensionality, the number of songs collected, and the scales of their features. To prepare these datasets for our playlist algorithm, we must account for these factors by transforming the data. We begin by removing any songs from our dataset that do not have features from all three audio datasets, leaving each dataset with 17,751 songs out of a possible 18,644. This results in the loss of 893 songs but we retain 95.2% of the Deezer dataset. Next, we apply a set of transformations to account for the dataset's varying dimensionality and feature scales.

6.1 Principal Component Analysis

Spotify and the MSD provide audio feature datasets that differ in dimensionality. More specifically, there are only 15 Spotify features compared to the 78 features of the MSD. Inputs with high dimensionality have typically hindered the performance of many approaches in machine learning, including nearest neighbor methods [43]. We examine the effect of this dimensionality in song transitions by comparing playlists made by the full Spotify and MSD features against those made by a reduction of these datasets.

To examine the impact of dimensions on audio features directly, we employ a Principal Component Analysis (PCA) to reduce the Spotify features, MSD features, and a combination of Spotify and MSD features to 12 dimensions each. Table 7 shows that this number explains over 95% of the variance in Spotify features, using an approach in line with common machine learning practices [64].

We employ the `PCA`[10] module from `scikit-learn` [54] for this transformation, which uses Singular Value Decomposition (SVD) to project the feature data into a lower-dimensional space. Figure 7 illustrates the effects of this PCA transformation on the Spotify features, showing that a PCA yields 12 features with zero correlation, or maximum covariance. Only musical features from Spotify and MSD are transformed with PCA, and the valence and arousal values from Deezer are untouched.

[10] scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA

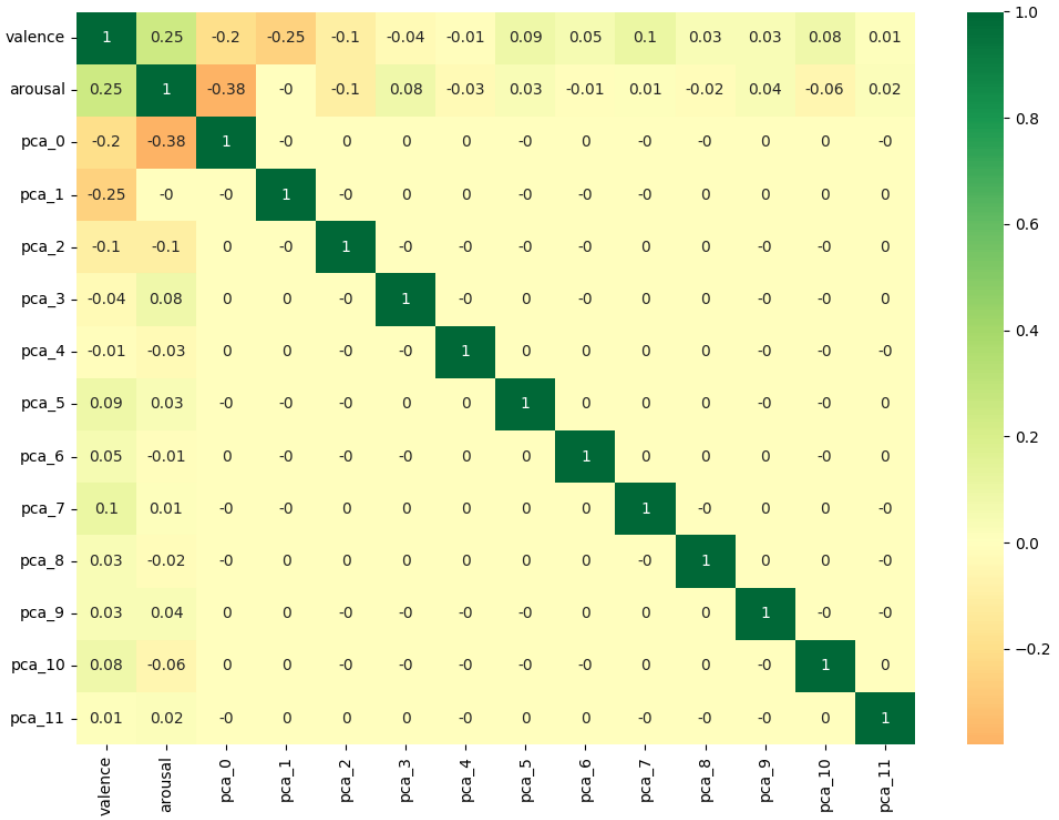


Figure 7: Correlation matrix after the PCA transformation.

6.2 Data Cleaning Process

Since we aim to create playlists that travel across various feature spaces, it is imperative to transform each feature to a consistent scale. Originally, we hypothesized that transforming data to have zero mean and unit variance, would suffice. However, we observed that transforming data using the `StandardScaler`[11] included with the `scikit-learn` Python package [54] still resulted in different tail sizes between features with large numbers of outliers, as depicted in Figure 8. This is a problem because features with larger or more extreme values would have an outsized effect on the path of the song.

[11] scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler

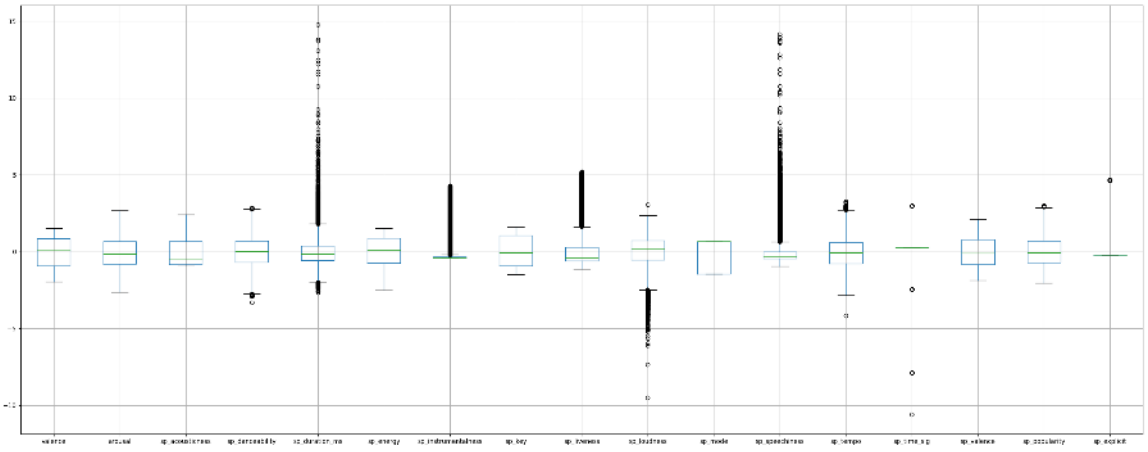


Figure 8: Deezer dataset and Spotify features after standardization. Notice the outliers on most features and the inconsistent ranges between features.

To reduce the effect of outliers, we employed a power transformation on each feature to rein in each distribution. To achieve this, we applied `scikit-learn`'s [54] `PowerTransformer`[12], which employs the Yeo-Johnson transformation [74], to reduce the range of outliers.

While this transformation managed most outliers to reasonable levels, some features still retained extreme outliers outside the interquartile range (IQR). We defined extreme outliers as any features that had values that strayed $5.5 \times IQR$ from the median. Features that contained extreme outliers were discretized into 20 bins using `scikit-learn`'s [54] `KBinsDiscretizer`[13]. We determined 5.5 as an appropriate coefficient for transforming extreme outliers across datasets since smaller coefficients (i.e. 2.5 or 3) would discretize almost every feature for some datasets, such as those with segment data. In our preliminary experiments, we found that a coefficient of 5.5 provided a balance between keeping the data continuous and removing the most extreme outliers.

After discretization, all features were normalized into a $[-1, 1]$ range, with a minimum value of -1 and a maximum value of $+1$, to keep the total distance for each feature the same and ensure no single feature is given undue weight in calculation for the transition between songs. We used the `MinMaxScaler`[14] from `scikit-learn` [54] to achieve this goal. Figure 9 presents the Deezer and Spotify features after transformation.

[12] scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PowerTransformer

[13] scikit-learn.org/stable/modules/generated/sklearn.preprocessing.KBinsDiscretizer

[14] scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler

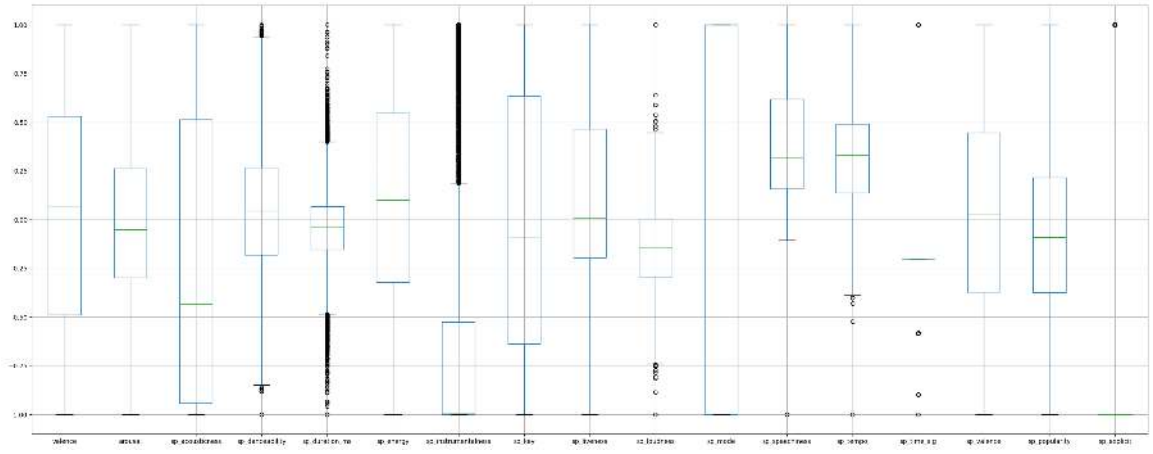


Figure 9: Deezer dataset and Spotify features after the power transformation, discretization, and normalization process.

We apply this transformation to all data, including the original Deezer valence and arousal data, Spotify features, MSD features, segment-based data, and PCA datasets.

6.3 Song Dataset Class

To manage mood, audio and segment-based features for 17,751 songs, we have built a custom class to contain our datasets. Using the `pandas` package in Python, we create a separate `DataFrame`, or table, for each audio feature dataset and the Deezer 2018 dataset, containing features for each song. For a segment-based dataset, we split the audio feature tables into two tables − one for the features for the head of the song and another for the tail.

For the purposes of this study, we will be generating large numbers of playlists to test the algorithm's capability. To save time when running our experiments, we also create an unsupervised `scikit-learn` [54] KNN model using the unique points of the Deezer 2018 dataset in our class, which is reused for creating every playlist. Finally, for the purposes of our algorithm in Section 7, we save a map which returns the Deezer song IDs that share a given Valence-Arousal point.

7 Mood-Dynamic Playlist Algorithm

In this work, we build upon existing approaches to create an algorithm that generates mood-dynamic playlists. Given an origin and destination song and a desired number of songs, our tool can generate a playlist in which the songs form a path between the origin and destination in the Valence-Arousal emotional space, with additional considerations given to the Spotify, MSD, or segment-based audio feature space.

Our playlist generator requires a few key input parameters. First, the user provides the algorithm with a dataset that contains Valence-Arousal data and audio features for songs. In addition, they specify an origin song S_1 and a destination song S_N, which must be IDs for songs that exist inside the dataset. Finally, the user specifies the number of songs N desired in the playlist.

In a different approach to Cardoso [10], our playlist algorithm works sequentially. choosing each subsequent song in a loop until the length L of the playlist is equal to $N-1$. Each iteration of this loop begins with the knowledge of the most recently chosen song, the destination song, and the number of songs remaining.

Within this loop lies a two-stage algorithm to select the next song, which we use to ensure a balance between emotional and genre transitions. This approach is similar to approaches outlined in Zamani [75], where the first stage of the algorithm recalls a large number of candidate points to be re-scored for accuracy in the second stage. Such a method is required since the high dimensionality in each music feature dataset could vastly overpower the two Valence and Arousal features in the Deezer dataset if not separated. For example, a combined KNN model with 15 Spotify features and two Valence-Arousal features would primarily choose neighbors based on audio characteristics. We describe our two-stage generator with pseudocode in Algorithm 1, and explain each stage fully in Sections 7.1 and 7.2.

Algorithm 1: Two-stage mood-dynamic playlist algorithm.

Input : Dataset D. KNN mood model K, and audio vectors V, $\forall\ S \in D$.
Origin and destination songs for the playlist $S_1, S_N \in D$.
Length of the final playlist N.

Output: A playlist P of N ordered songs $[S_1 ... S_N] \in D$.

$P, L \leftarrow [S_1], 1;$
while $L < N - 1$ **do**
  $[S_{candidates}] \leftarrow \text{neighbors}(K, S_L, S_N, L)$; /* Stage 1 */
  $[S_{candidates}] \leftarrow \text{filter}([S_{candidates}], P);$
  $S_{L+1} \leftarrow \text{distance}(V, [S_{candidates}], S_L, S_N, N - L)$; /* Stage 2 */
  $P, L \leftarrow [P, S_{L+1}], L + 1;$
end
$P \leftarrow [P, S_N];$

After the execution of the loop shown in Algorithm 1, the destination song is added to the playlist. This is returned as a `pandas` DataFrame table containing Deezer IDs for each song in the playlist, along with the Valence and Arousal points, artist and name for each song. Table 8 describes the songs in an example playlist that charts a path through the Valence-Arousal space shown in Figure 10, starting at a sad, depressed region and eventually ending at a happy, excited song.

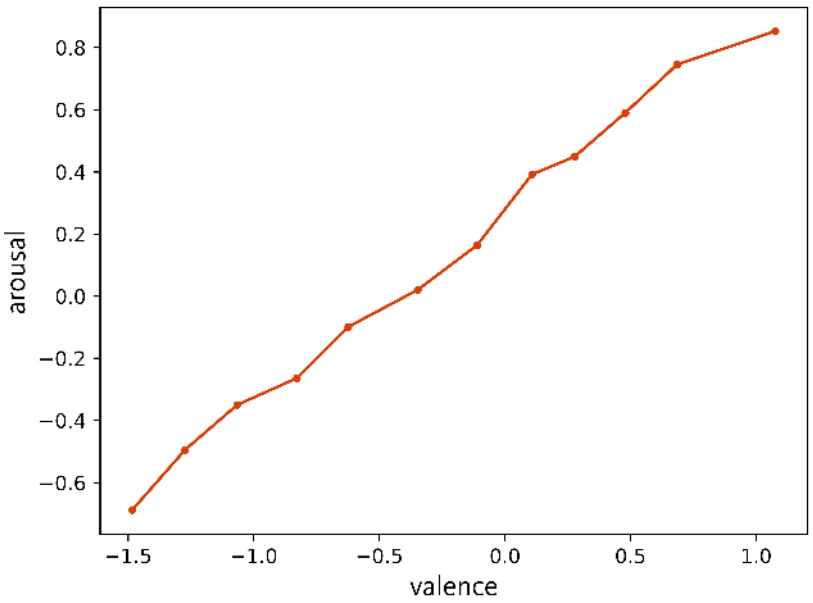


Figure 10: An example playlist traveling through the Valence-Arousal space. Each point is a song listed in Table 8.

Artist	**Song Title**	**Valence**	**Arousal**
Patty Loveless	How Can I Help You Say Goodbye	-1.483343	-0.688598
Elefant	Ester	-1.272664	-0.494736
Crossfade	Cold	-1.066284	-0.350541
Vanessa Carlton	Paradise	-0.827658	-0.264024
Julian Casablancas	Tourist	-0.623428	-0.099001
Aimee Mann	Going Through the Motions	-0.347825	+0.019559
Röyksopp	The Girl and The Robot	-0.110919	+0.163753
Black Kids	Love Me Already	+0.107500	+0.391261
Carpenters	Yesterday Once More	+0.277763	+0.448938
Interpol	C'mere	+0.478983	+0.589929
311	Mindspin	+0.685363	+0.745339
Rachael Yamagata	1963	+1.075764	+0.852684

Table 8: Artist, title, and points for the songs in the playlist shown in Figure 10.

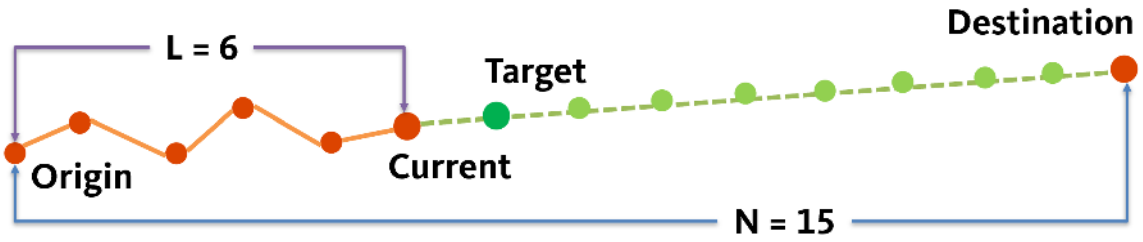


Figure 11: A diagram showing the target point as the first step in a linear trajectory from the current point in the playlist to the destination.

7.1 Stage 1: K-Nearest Neighbors

The first stage in the playlist loop uses a K-Nearest Neighbors (KNN) model to choose K points in the Valence-Arousal space that represent songs in our music emotion dataset, similar to approaches in Cardoso [10] and Flexer [25].

Given the current S_L and destination S_N songs as points in the Valence-Arousal space, and the number of loop iterations remaining $N-L$ to achieve the playlist length desired by the user, we calculate a hypothetical target point S_T using Equation (5). This equation divides the line between the current and destination songs into even steps in the Valence-Arousal space, and creates a hypothetical point as a single step down this path. Figure 11 visualizes the relationships between these three points, with the target point as the first step in an N song linear path from the current to the destination song.

$$S_T = S_L + \frac{S_N - S_L}{N - L} \tag{5}$$

We query the nearest songs in our dataset to our hypothetical target point to using `scikit-learn`'s `NearestNeighbor` model, fit to the unique Valence-Arousal points in our music emotion dataset [54]. Because of the two dimensions and 2762 unique points in the Deezer 2018 dataset, this algorithm uses a K-dimensional tree to store points for a search, achieving a computational cost of $O[DN\log(N)]$, or $O[N\log(N)]$ given the consistent use of $D = 2$ Valence-Arousal dimensions in our KNN stage. In this model, `scikit-learn` uses the Euclidean distance to determine the K nearest neighbors to a point.

We extract the LK nearest neighbors to the target point from this model, where L is the current length of the playlist. Occasionally, the `NearestNeighbors` model returns points already chosen for the playlist. To avoid repeating points in the Valence-Arousal space, we manually filter to receive the K nearest neighbors that have not been already chosen. We test various values for K in Section 9.3.

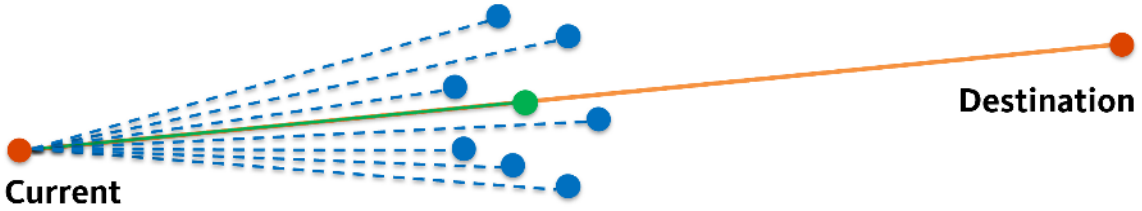


Figure 12: A visualization showing the candidate points and vectors (blue, dashed lines) which are compared to the target vector (green solid line).

7.2 Stage 2: Distance-based Song Selection

Stage 1 produces a set of K Valence-Arousal points representing songs that are candidates for the next song in our playlist. From these candidate points, we query our dataset for all of their corresponding Deezer songs and their acoustic features from an audio dataset. For datasets that contain separate features for the head and tail segments of each song, the features at the head, or the first few segments, of a song are returned.

When the current point nears the destination, one of the neighboring points from Stage 1 may be the destination point itself. This may happen if there are less than K points from the Deezer dataset between the current point and the destination in the Valence-Arousal space. In this case, one of the K points returned may be the destination point, which may contain the destination song as well as some songs that share that point. When the destination song is one of the candidates, our algorithm removes it to prevent the playlist from finishing early and having a length shorter than what the user requested.

Next, the audio features of the songs are then fed into Stage 2, which uses a distance metric configurable by the user to choose a single song. Here, we compare the distance between two vectors extending from the current song point: the vector from the current song to the candidate song, and a vector representing a single step on the line from the current song to the destination, calculated in a similar fashion to Equation (5). For segment-based data, we use the features at the tail of the current song to create a transition from the end of one song to the start of the next. Figure 12 demonstrates these points and vectors in two dimensions, with a vector to the target point as a step towards the destination in green, and blue vectors from the current song to each candidate being compared with various distance metrics.

These two vectors are compared using a distance metric which can be specified by the user. We test and compare various metrics in Section 9.1, including Cosine Similarity [58], Euclidean and Manhattan Distances [68], and Jaccard Distance [32]. The candidate song whose audio feature vector has the lowest distance score based on the metric specified is chosen as the next song. After Stage 2, the next song is

added to the playlist and becomes the current song to continue to the next iteration of the loop.

7.3 Time Complexity Analysis

Given a dataset of P unique Valence-Arousal points and D audio features, creating a playlist of N songs, we analyze our algorithm's computational complexity below. Before the loop runs, we construct a K-dimensional tree of the unique Valence-Arousal points by sorting each value in the two dimensions and splitting along their medians, with a complexity of $O[P\log(P)]$ [4]. During Stage 1 of the loop, we first select up to NK neighbors from the KNN model at a cost of $O[P\log(P)]$ as well.

For the algorithm's first stage, we filter up to NK neighboring points to a total of K songs, with a worst-case time complexity of $O[N^2K]$, since each candidate point is compared to N previous songs in the playlist. For the second stage, evaluating the songs for the K neighbors along D audio features takes a time complexity of $O[KD]$. Since this loop runs L times for the length of the playlist, and $P = 2,762$ unique points is far larger than the maximum values this experiment uses for $N(19), K(31), D(78)$, the total time complexity of the algorithm is $O[LP\log(P)]$.

8 Evaluation

We rigorously test our algorithm using a wide variety of parameters in order to better understand the quality of its playlists. These parameters include the K values for the algorithm's first stage, the distance metrics and audio datasets for the second stage, desired playlist lengths, and the number of segments in segment-based datasets.

8.1 Experimental Design

Our algorithm generates playlists in the Valence-Arousal feature space from the Deezer dataset, and is thus subject to its uneven distribution, as shown in Section 5.1. Therefore, a thorough test requires playlists that travel across various sections across this space. To ensure this diversity in travel, we create playlists that traverse between the four Cartesian quadrants in the Valence-Arousal space, in a similar approach to Deng [17]. For each quadrant in this space, we gather 100 Valence-Arousal points and choose a random song for each, resulting in 100 songs from each quadrant. The sample's distribution can be observed in Figure 13, with each point labeled and colored by quadrant.

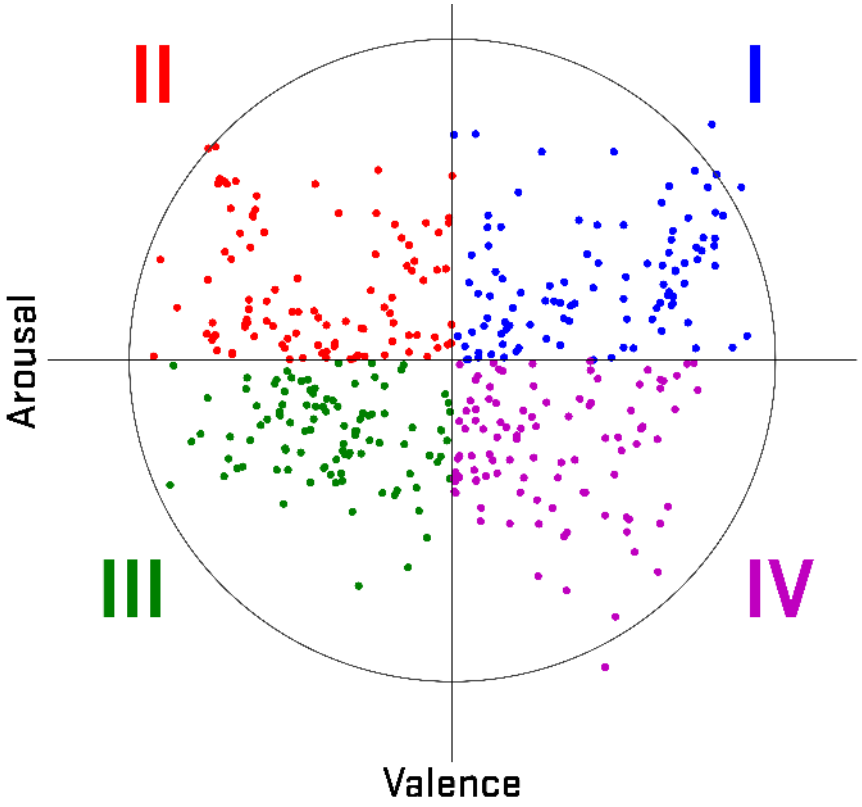


Figure 13: Map of the sample songs for each quadrant ($N = 100$ each), split by the Valence and Arousal axes, from the Deezer 2018 dataset [16].

Quadrants		Direction	Songs	Points
I	II	Horizontal	8448	1069
I	III	Diagonal	11407	1613
I	IV	Vertical	9236	902
II	III	Vertical	8515	1770
II	IV	Diagonal	6344	1059
III	IV	Horizontal	9303	1603

Table 9: The direction of travel, number of songs, and unique Valence-Arousal points for each pair of quadrants in the Deezer dataset.

We create playlists that traverse from one quadrant in the Valence-Arousal space to another. Table 9 shows the direction of travel and the number of songs and unique points in the Deezer dataset for each pair of quadrants. This table reveals uneven numbers of points between the pairs, which highlights the dataset's uneven distribution. For each of these six pairs, we build playlists from each point in the first quadrant to each point in the second quadrant, and vice versa. This yields 20,000 playlists for each of the six quadrant pairs, or 120,000 playlists total, for each dataset and parameter we test in any particular experiment.

8.2 Evaluation Metrics

Each song can be represented as a point (or vector) in the Valence-Arousal and audio feature spaces. Therefore, we evaluate the qualities of the playlist as a path of points from the origin to the destination. We adopt two key priorities in evaluating the quality of the playlists generated by our algorithm − smoothness and evenness − and create evaluation metrics for each.

8.2.1 Smoothness: Pearson Correlation Coefficient

First, the path should be *smooth*: the songs should fall in a linear path from the origin to the destination. This ensures that all of the emotional and audio characteristics of a point are consistently making changes from one song to the next. Nonlinear paths could result in playlists that change only in arousal for the first half of the playlist, and then abruptly change in valence while keeping a consistent energy level.

Figure 14 shows three 11-song playlist paths with different smoothness. The leftmost playlist (in red) has extreme divergence from the ideal line between the origin and destination, which is dotted. The middle playlist is smoother, with less variation from the ideal line, but the rightmost playlist stays closest to the ideal line and is thus the smoothest.

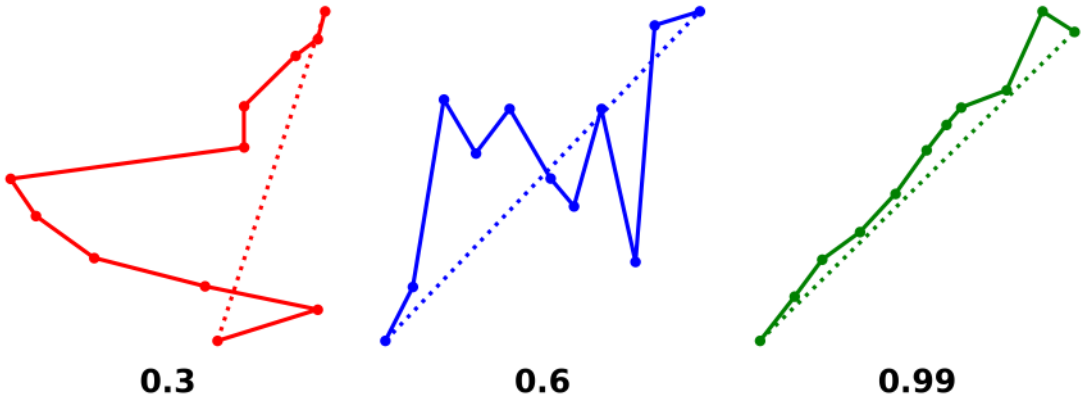


Figure 14: Example playlist paths with different Pearson Correlation Coefficients, representing playlists with poor, fair and strong smoothness respectively.

To evaluate a playlist's smoothness, we use a version of the Pearson Correlation Coefficient (PCC)[15]. Shown in Equation (6), this score measures how well two feature vectors A and B fall on a straight line on a score between 0 and 1, where 1 signifies perfect linear correlation. According to Rosen and Quadri, PCC has been shown to be an effective metric to evaluating the effectiveness of smoothing techniques [60].

$$\rho_{AB} = |\frac{\text{Cov}(A, B)}{\sigma_A \sigma_B}| \tag{6}$$

For the Valence-Arousal emotion space, we consider the PCC between valence and arousal to evaluate the smoothness of a playlist. For audio feature spaces, we calculate the average PCC for all pairs of features in the given space using a correlation matrix of the audio features given.

8.2.2 Evenness: Step Size Variance

Second, the path should be *evenly spaced*: there should be an equivalent jump in the emotional and audio characteristics from one song to the next. Uneven playlists could startle users in the case of an unusually large differences in the emotional or audio characteristics between two songs.

Figure 15 shows three paths of playlists with varying evenness, accompanied by a score showing the variance of these step sizes. The leftmost playlist (red) has a few emotional transitions that are much larger than others, seen by the longer line segments in the path. Comparatively, the middle playlist has a more even step size

[15] https://en.wikipedia.org/wiki/Pearson_correlation_coefficient

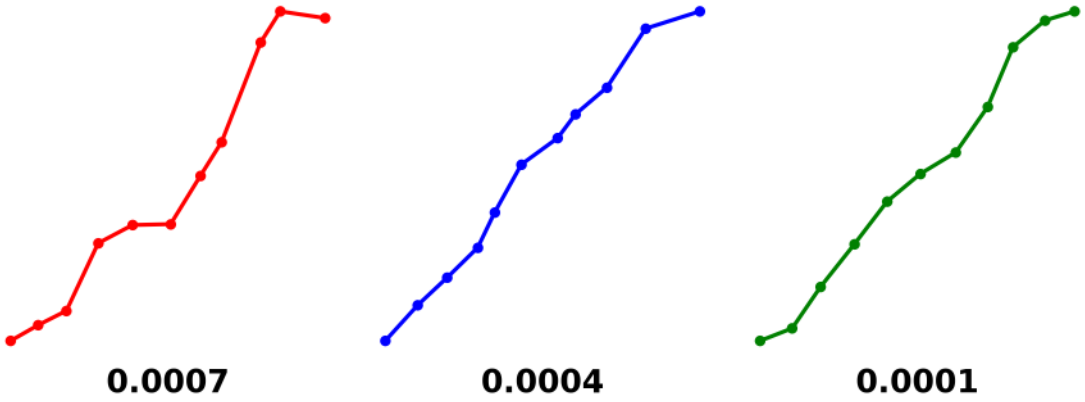


Figure 15: Example playlist paths with different step size variances, representing playlists with poor, fair and strong evenness respectively.

in the path, but the rightmost playlist is the most even, with all of the steps having roughly the same size.

To quantify the evenness of a playlist, we measure the variance of the distribution of step sizes between each pair of songs in the playlist. To find distances between adjacent songs S_k and S_{k+1} over D features, we calculate their Root Mean Square Error (RMSE), as shown in Equation (7).

$$RMSE_k = \sqrt{\frac{||S_{k+1} - S_k||_2}{N}} \tag{7}$$

To calculate the RMSE, we employ the Euclidean distance to find the distance between S_k and S_{k+1} and average these distances by the number of features to create a standard measure between feature sets of varying dimensionality. The RMSE is used as the step size between point transitions, such that their variance shows the difference between each transition's magnitude.

To understand our algorithm's behavior in mood and audio spaces, we calculate step size variances separately for both the Valence-Arousal emotion space and the various feature spaces of audio datasets.

9 Experiments & Results

We empirically test our playlist algorithm's behavior against a wide variety of tunable parameters: the distance metrics and audio feature datasets for Stage 2, the K values for Stage 1, desired playlist lengths, and different numbers of segments for segment-based datasets. When we are not testing these specific parameters, we use $K = 7$ for our algorithm's first stage, Euclidean distance for the second stage, and various audio feature datasets. By default, we generate 12-song playlists, since a 2014 survey of automatic playlist generation studies found that three datasets of playlists from popular sources contained 11.6 tracks on average [7].

9.1 Distance Metrics

We first test the effects of different common distance metrics on Stage 2 of our algorithm, examining their evenness and smoothness across audio and mood spaces. Here, we examine the impact of using the Euclidean, Manhattan, Cosine, and Jaccard distances to compare candidate songs, against a baseline method which randomly chooses a single neighbor.

Table 10 shows that the choice of distance metric for the audio features in Stage 2 has no significant effect on a playlist's smoothness or evenness in mood. Since the first stage chooses the K closest songs in the Valence-Arousal space to the target, the candidate songs are already highly similar in mood. This mitigates the effect of different distance metrics on these songs' audio features, which could choose the song that provides the best audio transition but the worst emotional transition. This finding informs developers and users that the choice of distance metric to optimize travel on audio features does not hurt a playlist's transitions on mood.

Table 11 reveals that using the Cosine, Euclidean and Manhattan distances for Stage 2 create the smoothest playlists in audio feature spaces, as measured by the PCC. The Jaccard distance performs poorly in comparison, performing similarly to the random neighbors baseline. Unlike the other three distance metrics, it does not

	Pearson Correlation			**Step Variance**		
	Mean	*St.Dev.*	*Median*	*Mean*	*St.Dev.*	*Median*
Cosine	0.928681	0.162085	0.988357	0.001140	0.001299	0.000749
Euclidean	0.929177	0.161056	0.988305	0.001143	0.001304	0.000750
Jaccard	0.928412	0.162658	0.988152	0.001149	0.001332	0.000744
Manhattan	0.929301	0.160845	0.988395	0.001132	0.001289	0.000745
Random	0.928845	0.161913	0.988449	0.001134	0.001303	0.000745

Table 10: Mood-based scores of playlists based on the distance metric in Stage 2.

Dataset	**All (Spotify + MSD)**			**Spotify**		
PCC	*Mean*	*St.Dev.*	*Median*	*Mean*	*St.Dev.*	*Median*
Cosine	**0.458472**	0.021189	**0.454996**	0.463596	0.045715	0.457373
Euclidean	0.456487	0.021741	0.452252	0.462629	0.048104	0.455204
Jaccard	0.454744	0.019066	0.451646	0.449793	0.040179	0.444198
Manhattan	0.457390	0.022524	0.452986	**0.464987**	0.049874	**0.457012**
Random	0.453964	0.018526	0.451040	0.447270	0.037652	0.442471

Table 11: Audio-based PCC of playlists by distance metric, measuring smoothness across two audio feature spaces.

consider the vector between two points, but only the minima and maxima of the two points' numerical values.

The use of the Cosine distance metric in Stage 2 yields the smoothest playlists in most audio feature spaces. Since the Pearson Correlation Coefficient is a measure of linearity, smoother playlists have songs that lie closer to a line. The Cosine distance measures the angle between candidate songs and a target point that falls in a perfect line. Minimizing this distance yields closer songs to that ideal line, not just the target point. This implies that the Cosine distance is an effective tool in creating smooth playlists. However, while this approach creates smoother playlists than Euclidean and Manhattan distances in a combined feature space, it fails to improve upon the Euclidean and Manhattan distances when only considering Spotify audio features. This may be a result of the skews of the distributions of features we observe in the Spotify and MSD feature spaces.

Figure 16 reveals that using Euclidean and Manhattan distances creates more evenly-spaced playlists than any other metric. While using the Cosine distance yields similarly smooth playlists to those using Euclidean and Manhattan distances, its singular consideration of the angle between two vectors means that it does not factor the magnitude of the vector between them. As a result, a song with a far larger or smaller step size than the target vector, but with a smaller angle to it, may minimize the Cosine distance. This can lead to more linear playlists but at the expense of an uneven step size.

Using the Euclidean and Manhattan distances yields songs with the shortest vectors to the target point at each loop in our algorithm. This approach yields songs that are the closest to the ideal target points, or playlists that are both smooth and even, because the target points fall at even steps in the line between the playlist's origin and destination songs. These results indicate that the Manhattan and Euclidean distances, also known as the L1 and L2 vector norms, are the strongest distance metrics for our algorithm's second stage.

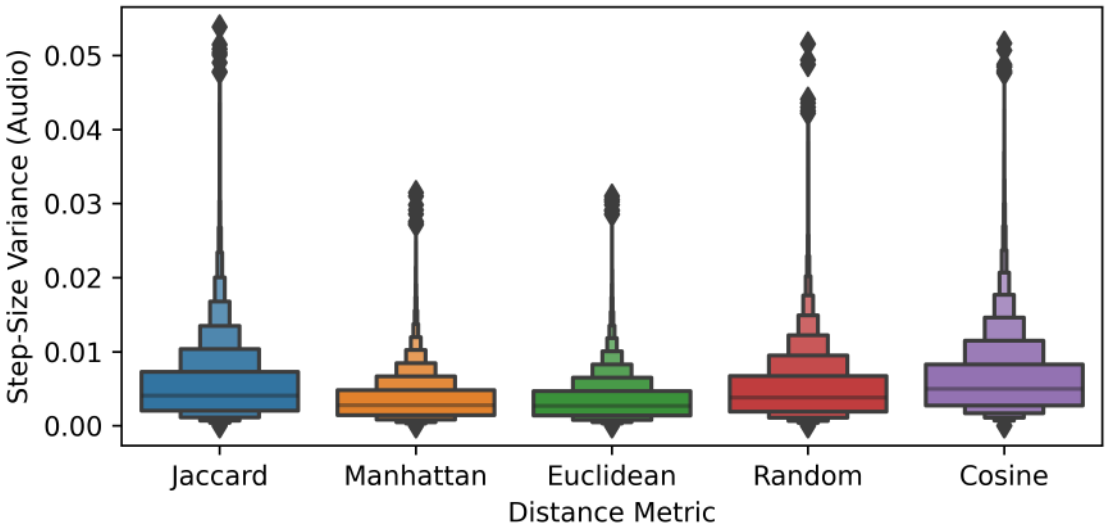


Figure 16: Step-size variances of playlists by distance metric, measuring evenness across an audio feature space.

9.2 Audio Datasets

We next examine the audio feature spaces to be traversed by our playlist algorithm in Stage 2. In this test, we compare the effectiveness of using *Spotify* features, *MSD* features, and a combined set of both (titled *All*) in our algorithm, against a baseline of reusing the mood features from the Deezer dataset. To inspect the impact of variance and dimensionality, we include 12-feature PCA datasets of the Spotify, MSD, and combined sets of features in our tests. We also juxtapose the aggregate audio feature datasets with two segment-based datasets, that collect average scores of features for the first and last 100 segments, and the first and last 30 seconds, of each song.

Table 12 displays the smoothness and evenness in the Valence-Arousal space of playlists created with various datasets. When evaluating emotion-based travel, using the Deezer features for both spaces leads to smoother and more even playlists based on mood. The choice in audio feature dataset has no impact on mood-based smoothness or evenness. This performance discrepancy is expected, since using only mood features effectively turns our algorithm into a 1-NN model, where the song emotionally closest to a target is chosen every time. By introducing separate audio feature spaces into the algorithm, we require playlists to balance emotional and audio features, which results in playlists that may be less smooth or even in mood transitions but provide songs that are more consistent in genre.

A closer inspection of the mood-based Pearson Correlation scores reveals a slightly more optimistic story for audio feature datasets. While the mean difference be-

	Pearson Correlation			Step Variance		
	Mean	*St.Dev.*	*Median*	*Mean*	*St.Dev.*	*Median*
Deezer	**0.933566**	0.159031	**0.990886**	**0.000311**	0.000302	**0.000224**
MSD	0.873421	0.205108	0.963398	0.001037	0.000869	0.000801
Spotify	0.873176	0.205091	0.962892	0.001043	0.000848	0.000816
All	0.873134	0.205579	0.963379	0.001040	0.000883	0.000796
PCA-MSD	0.871881	0.206737	0.962992	0.001040	0.000870	0.000805
PCA-Spotify	0.873293	0.205178	0.963092	0.001038	0.000852	0.000808
PCA-All	0.872370	0.205998	0.963027	0.001046	0.000864	0.000804
30 seconds	0.873470	0.205019	0.962998	0.001046	0.000876	0.000798
100 segments	0.873356	0.205373	0.963230	0.001051	0.000894	0.000804

Table 12: Mood-based scores of all playlists based on the dataset used in the second stage, with $N = 1,080,000$ playlists total ($120,000$ playlists per dataset).

tween mood-only playlists (those that use Valence-Arousal data for both stages) and playlists that use audio-based data for the second stage is quite large, the median smoothness scores are much closer. The left-hand side of Figure 17 visualizes this difference using box-and-whisker plots, showing the distribution of the PCC scores across playlists made with different audio datasets. In this plot, the boxplots indicate that over half of the playlists created in this scenario are almost perfectly linear in emotional travel, with PCC scores of over 0.96 (see Table 12), and the distribution's large tail implies that playlists of low smoothness in mood are the outlier. This reveals that using audio-based features for Stage 2 still yields playlists that travel smoothly in mood.

We observe that traveling through different pairs of quadrants of the Valence-Arousal space often results in varied performance. To provide an example, the right-hand side of Figure 17 isolates for playlists that travel from the third quadrant, with negative Valence and Arousal, to the first quadrant, with positive values for each. As a pair, these two quadrants induce diagonal travel in playlists and contain the most songs and a high number of unique Valence-Arousal points, as shown in Table 9. In this subset, playlists generated by audio are much smoother than the entire set, and their PCC scores are much closer to those using Deezer data for both stages. The variance of density is a result of the Deezer dataset's uneven distribution, and implies that the smoothness of mood-dynamic playlists is dependent on the distribution of points across the Valence-Arousal plane.

We measure the performance of each dataset in each audio feature space that we test, to evaluate the merits of different audio datasets and PCA across various audio feature spaces. Table 13 shows playlists' performance by dataset when evaluated against a combined Spotify and MSD feature space. Here, the playlists created

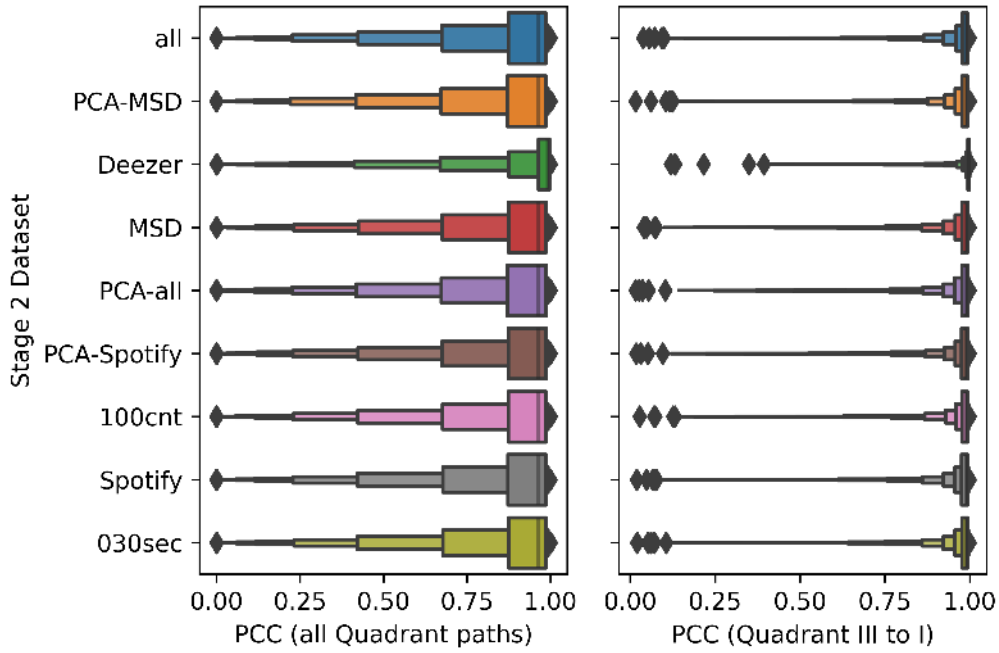


Figure 17: Smoothness of playlists by dataset, for all 1,080,000 playlists (left), and 90,000 playlists that travel only from Quadrant III to Quadrant I (right).

using MSD-based features (including the Spotify-MSD combined feature space) outperformed not only the Deezer dataset but all other audio datasets as well. This implies that the combined dataset was far more influenced by the 78 features of the MSD data than the 15 features from Spotify, and that the different audio datasets describing features of the same songs are not highly correlated with each other.

More interestingly, we observe in Table 13 that playlists were only smoother when applying PCA to MSD-related data, yet this came at the expense of a playlist's evenness. However, when evaluating against Spotify features alone, as shown in Figure 18, our algorithm generates smoother playlists when considering the Spotify dataset with or without PCA applied. In this feature space, using the combined dataset also yields much more even playlists than other datasets, second only to the Spotify features themselves.

These findings reveal a few key factors in the smoothness and evenness of playlists across audio features. First, the Spotify and MSD features share little correlation, and that segment-based datasets fail to yield improvements in these spaces compared to the baseline of purely mood-based playlists. In addition, using audio datasets with lower dimensionality yields smoother playlists, since applying PCA to 12 features yielded strong improvements over a 78-feature MSD dataset, but only mild improvements over the 15-feature Spotify data. However, while applying a PCA

	Pearson Correlation			Step Variance		
	Mean	*St.Dev.*	*Median*	*Mean*	*St.Dev.*	*Median*
Deezer	0.294501	0.014783	0.293004	0.006281	0.002941	0.005821
MSD	0.294415	0.018481	0.291252	**0.004019**	0.001832	**0.003746**
Spotify	0.292482	0.016099	0.290243	0.005449	0.002574	0.005030
All	0.294538	0.018706	0.291339	**0.003500**	0.001625	**0.003247**
PCA-MSD	**0.300396**	0.017297	**0.298441**	0.005870	0.002734	0.005438
PCA-Spotify	0.292421	0.015304	0.290586	0.005851	0.002665	0.005454
PCA-All	**0.299186**	0.017052	**0.297231**	0.005800	0.002692	0.005371
30 seconds	0.292375	0.014951	0.290564	0.005828	0.002736	0.005391
100 segments	0.293484	0.015424	0.291664	0.005857	0.002724	0.005418

Table 13: Audio-based scores of N =1,080,000 playlists based on dataset, scored in a combined Spotify and MSD feature space.

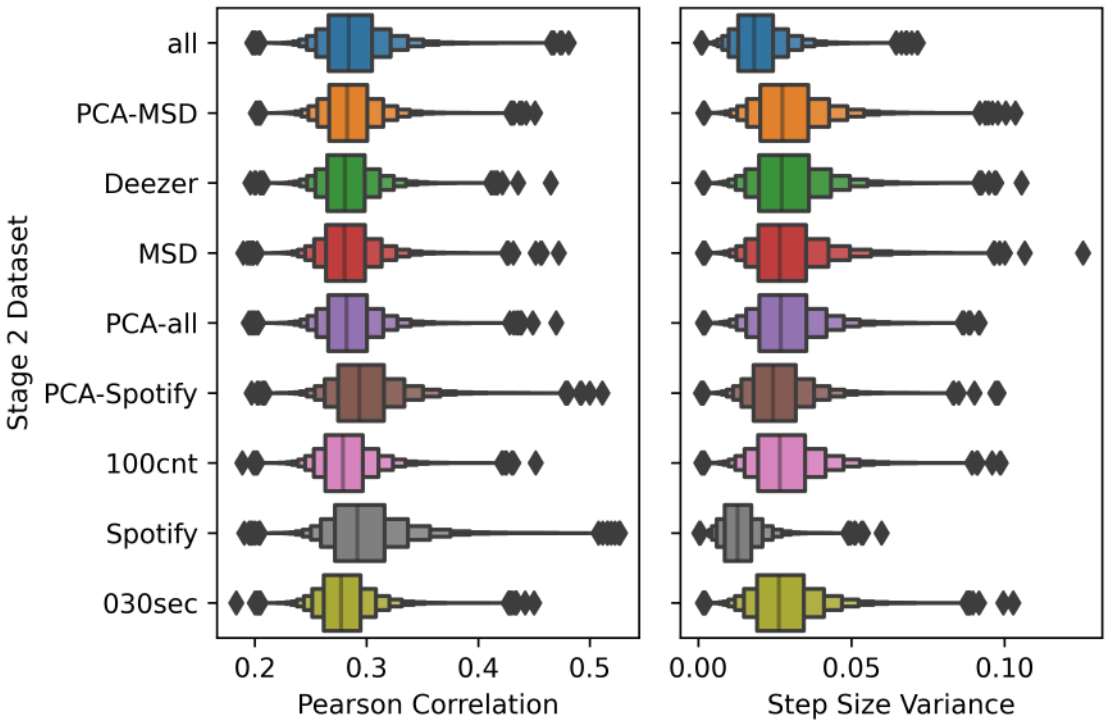


Figure 18: Smoothness (left) and evenness (right) scores of playlists based on dataset, scored in a Spotify feature space.

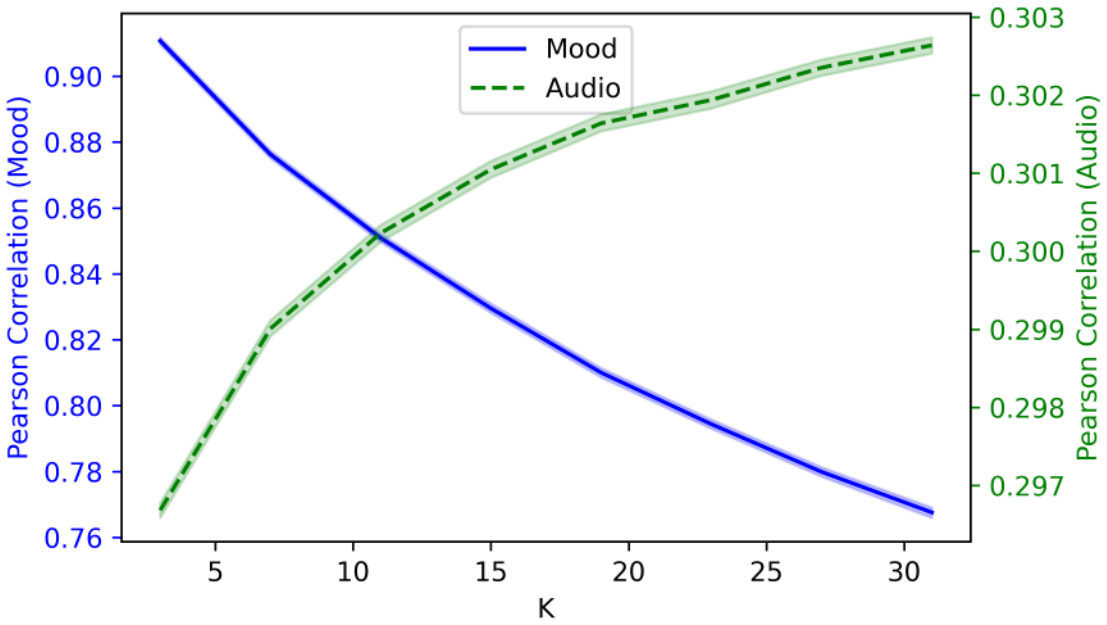


Figure 19: Pearson Correlations of playlists by K value, measuring smoothness across mood and audio spaces.

transformation can can retain the relative differences between songs, it loses their true differences, and sacrifices even transitions when evaluating against the original feature space.

9.3 K Values

In our third experiment, we evaluate different values for K, or the number of candidate points our algorithm extracts for a transition in the Valence-Arousal space. We examine the effects of extracting $K = [3, 7, 11, 15, 19, 23, 27, 31]$ points from our unsupervised KNN model in Stage 1 of our algorithm.

From Figures 19 and 20, we observe that as K increases, we generate playlists that travel more smoothly and evenly in audio feature spaces, at the expense of the smoothness and evenness in emotional travel. When considering a smaller K, recalling the K closest points means that these candidates are closer on average to the ideal Valence-Arousal transition. This enable our algorithm to create more precise emotional travel. However, yielding fewer songs in Stage 1 means that there are fewer choices of songs from which to optimize audio travel in Stage 2. A larger K reverses this effect, allowing the second stage to choose a song that provides the best audio travel, but this song could potentially be much farther than other candidates from the ideal emotional target. By first recalling a K songs based on mood transitions

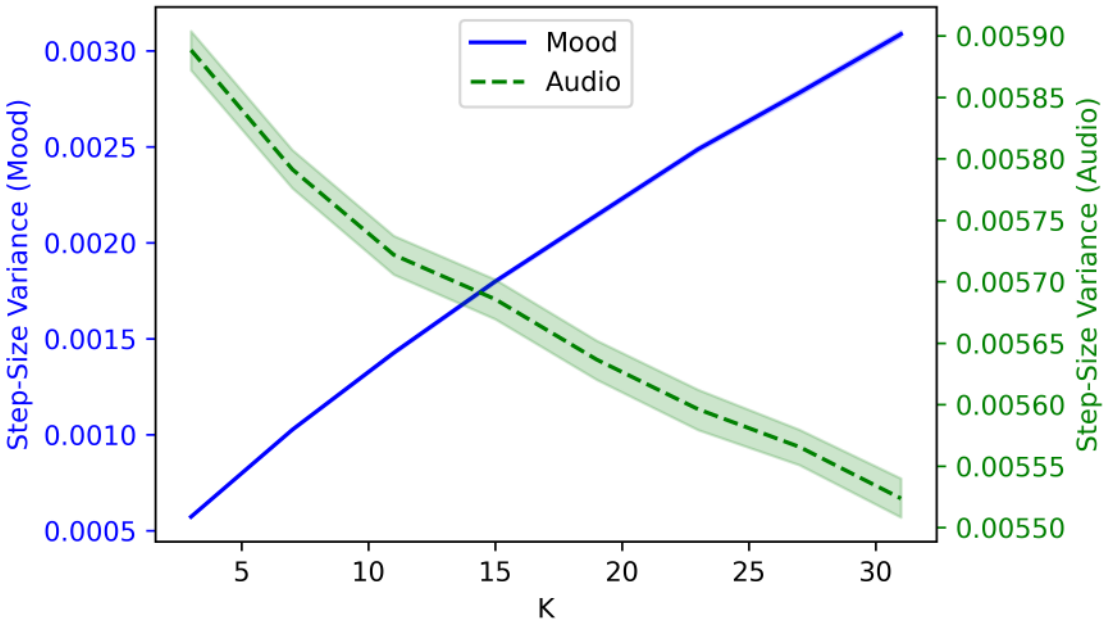


Figure 20: Step-Size Variances of playlists by K value, measuring evenness across mood and audio spaces.

and re-scoring for audio similarity, our two-stage algorithm gives designers and users the power to directly tune the balance between the emotional and genre transitions of their playlists.

9.4 Playlist Lengths

We also examine the effect of using different lengths to travel between the same songs as input for our playlist algorithm. Given the same origin and destination songs, as the length of the playlist between them increases, we expect the step sizes between songs to shrink. This enables us to study how the density of our dataset impacts playlists as the step size decreases. In the test, we examine playlists of $N = 3, 5, 7, 11, 13, 15, 17, 19$ songs. Figure 21 and Table 14 show the effects of different playlist lengths on the smoothness and evenness of playlists across both types of feature spaces.

As a playlist's length increases, the smoothness over both mood and audio spaces decreases. This effect is strongest when increasing from three songs to five, but weakens as playlists get longer, especially for the audio features in Stage 2. With larger playlists, the number of songs in between a fixed origin and destination increases, and so the step size for choosing the next song decreases. In this smaller scale, the distances between candidate song points in both feature spaces is much larger relative to the step size, and some regions may be sparse in their song points.

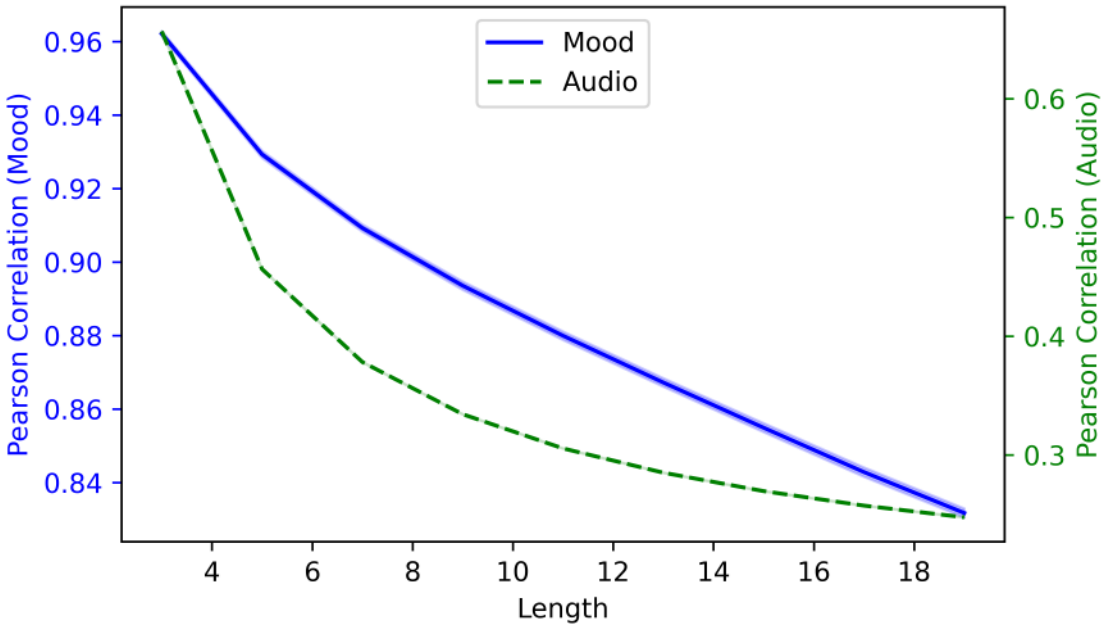


Figure 21: Pearson Correlations of playlists of different lengths, measuring smoothness across mood and audio spaces.

Since our algorithm recalculates the trajectory to the destination point when choosing each song, the choice of a song far from the direct line between the origin and destination has a disproportionate impact on the shape of the playlist path. In this scenario, the playlist will create a new line to the destination instead of returning to the direct line from the origin, and subsequent points will continue to be far from the ideal line. A longer playlist can lead to more opportunities for such behavior from our algorithm, potentially resulting in curved or jagged playlist paths that are not smooth in a linear sense.

Each subsequent song added to the playlist seems to consistently impact the smoothness of playlists, as seen in the linear decay in the mood-based PCC score in Figure 21. Over audio feature spaces, longer playlists experience a sharp decline, as their PCC score drops from over 0.6 with a three-song playlist to well below 0.3 for a 19-song playlist in Figure 21. However, this decay seems exponential, and seems to stabilize as the playlist continues to get larger.

Table 14 shows that three-song playlists are much more evenly-spaced than longer playlists, since their median step size variance is far smaller than all longer playlists. This is an interesting edge case of our algorithm, since a three-song playlist only runs the algorithm once to choose a single song between the origin and destination. These playlists also have a denser pool of candidate points relative to their large step

N	Mood-Based Evenness			Audio-Based Evenness		
	Mean	*St.Dev.*	*Median*	*Mean*	*St.Dev.*	*Median*
3	**0.000873**	0.001489	**0.000388**	0.003130	**0.004302**	**0.001458**
5	0.001142	0.001302	0.000750	0.003424	0.002799	0.002705
7	**0.001158**	0.001170	0.000818	0.003464	0.002244	0.002992
9	0.001112	0.001028	**0.000824**	0.003477	0.001910	0.003127
11	0.001061	0.000924	0.000804	0.003490	0.001706	0.003214
13	0.001011	0.000834	0.000785	0.003514	0.001559	0.003276
15	0.000965	0.000763	0.000765	0.003520	0.001443	0.003318
17	0.000925	0.000711	0.000743	0.003528	0.001341	0.003353
19	0.000888	0.000668	0.000722	0.003536	0.001264	0.003379

Table 14: Evenness of playlists by length across both mood and audio spaces.

size. However, the standard deviations of their step size variances, especially in audio features, is much larger than with longer playlists. This suggests that three-song playlists may often have more evenly-spaced playlists than longer playlists, but that this is highly dependent on the uneven density of the Valence-Arousal space in the Deezer dataset.

Outside of this edge case, Table 14 shows that playlist songs are less evenly spaced in their audio features as their length increases. This behavior is also consistent with mood-based evenness, though this effect reverses with playlists longer than seven or nine songs. Due to the inclusion of discretization for some audio features in our data cleaning process, some features have a step size of 0.1 between their 20 bins with values ranging from -1 to $+1$. For a longer playlist, this step in a feature becomes quite large relative to the ideal step size between songs, and may have an undue weight in song selection. The Valence-Arousal features from the Deezer dataset do not face this issue, as they were not discretized during the cleaning process. This suggests that discrete features should be avoided in audio datasets for dynamic playlist generation.

9.5 Dataset Segment Counts

The number of segments from the start and end of a song to include is configurable when compiling the weighted average of features for a segment-based dataset. Since these segments correspond to a single note or sound in a song, they do not have consistent duration. We account for this when we collect our segment datasets, so each song's features are a weighted average of a different number of segments at its head and tail. As a result, we compare nine datasets that are created from each song's first and last $D = [1, 2, 5, 10, 20, 30, 40, 50, 60]$ seconds. Each playlist is evaluated against the audio feature space of the segment dataset collected with $D = 30$ seconds, with

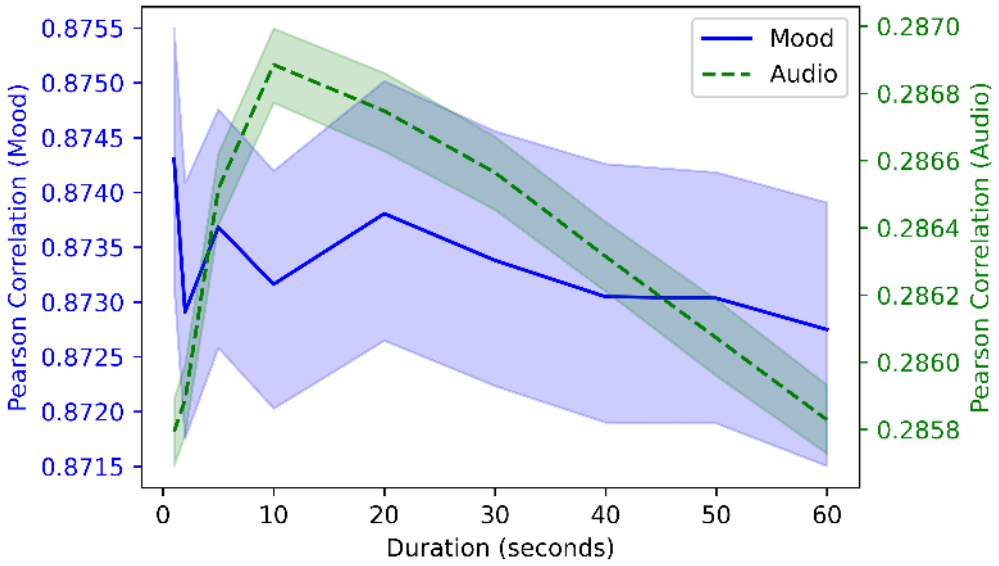


Figure 22: Pearson Correlation Coefficients of playlists generated from the first and last D seconds of songs across both mood and audio spaces.

results for both mood and audio in Figures 22 and 23.

In the Valence-Arousal space, there is no observable relationship between segment-based datasets of various durations, which is consistent with other audio-based datasets, as seen in Section 9.2. For a segment-based audio feature space, playlist smoothness increases with segment duration up until 10 seconds, after which it decreases. This peak at around 10 seconds of segments implies that the first and last 10 seconds are the most representative of a song's audio features, and including more seconds in the calculation with Equation (4) may hurt the smoothness of track transitions in dynamic playlists. This implies that studies in segment-based playlist generation can focus on optimizing the transitions between the last 10 seconds of one song and the first 10 seconds of the next to guarantee the smoothest playlists.

Datasets that used durations closer to 30 seconds generated more even playlists in a 30-second segment feature space. This is partially because a 30-second segment feature space was used in evaluation. Yet, from the relatively similar evenness for durations longer than 30 seconds, we see further evidence of the stabilization effect of Equation (4). Also shown in Figure 6, this shows that as more segments are included, the average changes more slowly, yielding diminishing returns in the evenness of dynamic playlists.

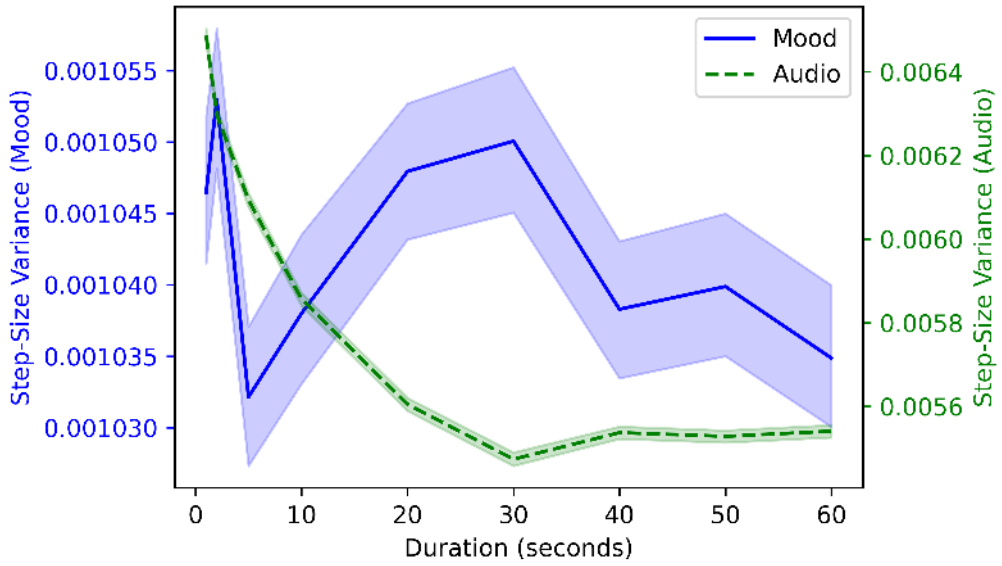


Figure 23: Step-Size Variances of playlists generated from the first and last D seconds of songs across both mood and audio spaces.

9.6 Regional Variances

Across all tests, the uneven distribution of points in the Valence-Arousal space has an effect on playlists' smoothness and evenness. Table 15 shows that opposite quadrant pairs, which induce diagonal travel, have the smoothest playlists overall. Within the same direction of travel, quadrant pairs with more unique points in the Valence-Arousal space had smoother playlists. These findings indicate that playlists which transition in both the Valence and Arousal of their songs are likely to provide a smoother listening experience for users. Transitioning in both emotion dimensions represents potentially the most common and effective use case of dynamic playlists, where a user may need to be cheered up after feeling sad and depressed, or perhaps to find calm after feeling tense and angry.

Our results also indicate that more points in the origin and destination quadrants can also improve a playlist's evenness. In our algorithm, a denser region in the Valence-Arousal space means that the K nearest candidates to an ideal target point are closer than they would be in a sparser region. This means that the step size between subsequent songs will be closer to a consistent ideal, minimizing such variance. Thus, the distribution and density of Valence-Arousal points is a limitation of the Deezer dataset. To ensure playlists have evenly spaced songs and that the emotional transition is consistent for users in the listening experience, datasets should include a dense, even distribution of emotional points.

Quadrants				**Pearson Score**		**Step Variance**	
From	*To*	**Direction**	**Points**	*Mean*	*St.Dev.*	*Mean*	*St.Dev.*
I	II	Horizontal	1069	0.79463	0.24625	0.00145	0.00121
I	III	**Diagonal**	1613	**0.97196**	0.06451	0.00082	0.00060
I	IV	Vertical	902	0.83339	0.22720	0.00116	0.00086
II	I	Horizontal	1069	0.79614	0.24571	0.00144	0.00115
II	III	Vertical	**1770**	0.88879	0.19127	**0.00051**	0.00036
II	IV	**Diagonal**	1059	**0.95523**	0.10692	0.00092	0.00077
III	I	**Diagonal**	1613	**0.97432**	0.06047	0.00093	0.00072
III	II	Vertical	**1770**	0.88970	0.19305	**0.00056**	0.00043
III	IV	Horizontal	1603	0.83449	0.23158	0.00086	0.00086
IV	I	Vertical	902	0.83188	0.22929	0.00120	0.00089
IV	II	**Diagonal**	1059	**0.95342**	0.10865	0.00088	0.00065
IV	III	Horizontal	1603	0.83288	0.23052	0.00075	0.00069

Table 15: Mood-based scores for playlists ($N = 1,080,000$) generated during the dataset test, separated by the origin and destination quadrants of the playlists.

10 Discussion

Previous studies have introduced methods to create playlists that are dynamic in either mood [10] or audio spaces [25]. We combine these aspects into a multi-stage algorithm that generates playlists that gradually travel through both spaces between origin and destination songs. Our novel approach creates a path through vector spaces of song features, and thus requires no personalized data on existing playlists or the status of the user. We define numerical metrics for a playlist's smoothness and evenness, and a novel testing methodology for mood-dynamic playlists which evaluates across several regions of a feature space. This is the first approach in the automatic playlist generation field that evaluates dynamic playlists as paths in a vector space of mood and audio.

10.1 Summary of Findings

We observe that our algorithm's input parameters have a strong effect on the performance characteristics of playlists. The balance of a playlist's performance between mood and audio characteristics can be easily tuned by adjusting the recall of its first stage. Our approach builds upon research into two-stage algorithms [75] by using the high recall of the first stage as an important parameter.

In the algorithm's second stage, using the Cosine distance metric and reducing the dimensionality of feature data can lead to slightly smoother travel through audio. This ensures that each song in a playlist moves the user on a linear emotional and acoustic journey. However, this comes at the expense of consistent steps through these spaces, potentially making these transitions more jarring for users.

In addition, the distribution of Valence-Arousal points in our dataset has a strong effect on a playlist's smoothness or evenness. Controlling for the first and final songs of the playlist, including more songs in between reduces the size of each emotional transition between tracks. Requesting shorter playlists, with a larger step between songs, leads to smoother playlists than a smaller step due to the relative sparsity of the data at a smaller scale.

Playlists that travel through sparser regions of our Valence-Arousal feature space tend to vary more in the magnitude of their emotional transitions from one song to the next. However, smoothness is more dependent on the direction of travel; playlists that transition heavily in both Valence and Arousal more likely to have their points lie on a linear path than those which only mainly travel in one dimension.

10.2 Limitations

The primary limitation of our work is the quantity and quality of music emotion datasets. Although the Deezer 2018 dataset [16] is a large repository of songs and their Valence-Arousal scores, the synthetic nature of its emotion annotations limits their effectiveness. As discussed earlier in Section 5.1, as many as 1700 songs share a Valence-Arousal point, and the number of unique points is only 15% of the total number of songs. This required us to implement additional structures that filter for unique points and handle them, as described in Section 6.3.

Datasets that collect Valence-Arousal readings from humans not only potentially serve as a more accurate reflection of a song's mood, but can avoid this problem due to the variance of multiple users labeling songs. However, existing datasets that employ these methods [1, 11, 76] are too small in size and sometimes consist of less popular songs for which it is more difficult to find other features from public datasets.

10.3 Future Work

Building on the contributions of this work, there are many opportunities for future improvements and extensions of our mood-dynamic playlist algorithm. Currently, we evaluate a purely linear trajectory from an origin to a destination song, but some approaches allow users to draw a curved path on the Valence-Arousal plane on which to generate playlists [10, 14]. Adding a curved trajectory for our playlist's song selection could potentially allow for more personalized transitions for users, but this requires testing to see if it creates a more satisfying listening experience.

In this experiment, we collect data on the segments of a song and demonstrate that they have little effect in improving playlists. However, our current evaluation methods only examine smoothness and evenness at the whole song level, treating each song as an acoustic and emotional monolith. Future studies can examine the change in mood throughout the duration of a song, and perhaps create a smoother listening experience for the user. Furthermore, this approach can be extended to personalized data, such as a user's ratings of songs or listening history, to create playlists that directly reflect their own specific taste in music.

This works presents an opportunity to test the effectiveness of mood-dynamic playlists on human subjects to measure the change in a user's emotion throughout the listening experience. Previous studies in dynamic playlists [25, 10] either do not evaluate their recommendations or use existing data to measure playlists which travel in genre, but not in mood. Our algorithm can power an application that recommends a playlist based on the user's current mood a state which they desire. During and after listening to the playlist, the user's mood can be measured to examine if the desired emotional transition from the playlist was achieved.

10.4 Applications

Researchers could potentially integrate this algorithm with a human-in-the-loop system that collects the user's current mood. A few studies [19, 66] explore systems for recognizing emotions in facial expressions, and recommending music that matches the user's mood. Other studies use data from accelerometers [35, 21] or electrocardiograms [29] to choose songs related to a user's present state. However, such system can also factor context-related data such as foreground computer processes [65] and the current climate [29, 59] when recommending songs to users.

However, with the exception of Liu et al. [40] whose approach seeks to lower a user's high heart rate with calming music and vice versa, most studies seek to keep the user in the same mood. Future studies might integrate these methods for emotion recognition with our work in experiments that examine the change in a listener's mood after listening to a mood-dynamic playlist generated by our algorithm.

This playlist algorithm could also be integrated into continuous systems which recommend the next song to play based on user preferences and a desired target mood. Studies from Chi [13] and Liebman [39] use reinforcement learning techniques that repeatedly adjust recommendations based on whether a user replayed, skipped, or rated the current song. With our algorithm, such a system could change recommendations based on the user's current mood, perhaps as an extension to recent studies such as Rumiantcev [61] which allow users to report their current mood and environment to deliver them their next song. We show that a three-song playlist, where the algorithm just chooses a single step towards a desired mood, has strong performance. However, this requires more thorough examination on the proper step size to use for the algorithm, and user testing to evaluate the true emotional impact of such playlists.

10.5 Conclusion

This work presents a novel and efficient exemplar-based learning algorithm leveraging existing public datasets of songs to create musical playlists that travel in mood while maintaining similarity in audio. We define and measure two key qualities in evaluating a playlist's effectiveness, utilizing the Pearson Correlation Coefficient and Root Mean Square Error to measure smoothness and evenness in continuous mood and audio spaces. Our experiments indicate that this algorithm can be tuned through several parameters to adjust the effectiveness of a playlist's track transitions. To our knowledge, this is the first study that uses a nearest-neighbor approach to construct and evaluate mood-dynamic playlists that also smoothly transitions by acoustic similarity. Such an interactive tool could serve as an aid to music therapy, gently nudging users from their current mood towards more happy, focused, or content state of mind.

References

[1] Anna Aljanaki, Yi-Hsuan Yang, and Mohammad Soleymani. "Developing a benchmark for emotional analysis of music". In: *PLOS ONE* 12.3 (Mar. 2017), pp. 1–22. DOI: 10.1371/journal.pone.0173392.

[2] Alo Allik, Florian Thalmann, and Mark Sandler. "MusicLynx: Exploring music through artist similarity graphs". In: *Proceedings of the The Web Conference*. 2018, pp. 167–170. DOI: 10.1145/3184558.3186970.

[3] N. S. Altman. "An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression". In: *The American Statistician* 46.3 (1992), pp. 175–185. DOI: 10.1080/00031305.1992.10475879.

[4] Jon Louis Bentley. "Multidimensional Binary Search Trees Used for Associative Searching". In: *Commun. ACM* 18.9 (Sept. 1975), pp. 509–517. ISSN: 0001-0782. DOI: 10.1145/361002.361007.

[5] Thierry Bertin-Mahieux et al. "The Million Song Dataset". In: *Proceedings of the 12th International Conference on Music Information Retrieval*. ISMIR 2011. 2011, pp. 591–596. DOI: 10.7916/D8NZ8J07.

[6] Deborah J Blood and Stephen J Ferriss. "Effects of background music on anxiety, satisfaction with communication, and productivity". In: *Psychological reports* 72.1 (1993), pp. 171–177. DOI: 10.2466/pr0.1993.72.1.171.

[7] Geoffray Bonnin and Dietmar Jannach. "Automated Generation of Music Playlists: Survey and Experiments". In: *ACM Computing Surveys* 47.2 (Nov. 2014). ISSN: 0360-0300. DOI: 10.1145/2652481.

[8] Théo Bontempelli et al. "Flow Moods: Recommending Music by Moods on Deezer". In: *Proceedings of the 16th ACM Conference on Recommender Systems*. RecSys 2022. Seattle, WA, USA: Association for Computing Machinery, 2022, pp. 452–455. ISBN: 9781450392785. DOI: 10.1145/3523227.3547378.

[9] Emilios Cambouropoulos and Maximos Kaliakatsos-Papakostas. "Symbolic Approaches and Methods for Analyzing Musical Similarity: Representation and Pattern Processing in Harmony". In: *The Oxford Handbook of Music and Corpus Studies*. Oxford University Press. ISBN: 978-0-19-094544-2. DOI: 10.1093/oxfordhb/9780190945442.013.9.

[10] Luís Cardoso, Renato Panda, and Rui Pedro Paiva. "MOODetector: A Prototype Software Tool for Mood-based Playlist Generation". In: *Proceedings of the 2nd Portuguese National Symposium on Informatics*. INForum 2011. Sept. 2011.

[11] Yu-An Chen et al. "The AMG1608 dataset for music emotion recognition". In: *Proceedings of the 40th IEEE International Conference on Acoustics, Speech and Signal Processing*. ICASSP 2015. 2015, pp. 693–697. DOI: 10.1109/ICASSP.2015.7178058.

[12] Shuo Chen et al. "Playlist prediction via metric embedding". In: *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery*

and Data Mining. Beijing China: ACM, Aug. 2012, pp. 714–722. DOI: 10.1145/2339530.2339643.

[13] Chung-Yi Chi et al. "A Reinforcement Learning Approach to Emotion-based Automatic Playlist Generation". In: *Proceedings of the 15th International Conference on Technologies and Applications of Artificial Intelligence*. TAAI 2010. Dec. 2010, pp. 60–65. DOI: 10.1109/TAAI.2010.21.

[14] Rahul Kashinathrao Dahule and Shubhangi Mahadeo Jadhav. "Music track exploration and playlist creation". US20140052731A1. Feb. 2014. URL: https://patents.google.com/patent/US20140052731/en.

[15] Hideo Daikoku et al. "Agreement among human and automated estimates of similarity in a global music sample". In: *Proceedings of the 10th International Workshop on Folk Music Analysis*. FMA 2022. PsyArXiv, July 2020. DOI: 10.31234/osf.io/76fmq.

[16] Rémi Delbouys et al. "Music mood detection based on audio and lyrics with deep neural net". In: *Proceedings of the 19th International Conference on Music Information Retrieval*. Ed. by Emilia Gómez et al. ISMIR 2018. 2018. DOI: 10.48550/arXiv.1809.07276.

[17] James J. Deng and Clement Leung. "Emotion-based music recommendation using audio features and user playlist". In: *6th International Conference on New Trends in Information Science, Service Science and Data Mining*. ISSDM 2012. Oct. 2012, pp. 796–801.

[18] Renee Donohoe and Teresa McNeely. *The Effect of Student Music Choice on Writing Productivity*. Tech. rep. US Department of Education, 1999.

[19] Anukriti Dureha. "An Accurate Algorithm for Generating a Music Playlist based on Facial Expressions". In: *International Journal of Computer Applications* 100.9 (Aug. 2014), pp. 33–39. ISSN: 09758887. DOI: 10.5120/17557-8163.

[20] Paul Ekman. "Facial Expressions of Emotion: New Findings, New Questions". In: *Psychological Science* 3.1 (1992), pp. 34–38. DOI: 10.1111/j.1467-9280.1992.tb00253.x.

[21] Greg T. Elliott and Bill Tomlinson. "PersonalSoundtrack: context-aware playlists that adapt to user pace". In: *CHI 2006 Extended Abstracts on Human Factors in Computing Systems*. CHI EA 2006. New York, NY, USA: Association for Computing Machinery, Apr. 2006, pp. 736–741. ISBN: 978-1-59593-298-3. DOI: 10.1145/1125451.1125599.

[22] Jeff Ens and Philippe Pasquier. "Quantifying Musical Style: Ranking Symbolic Music based on Similarity to a Style". In: *Proceedings of the 20th International Symposium on Music Information Retrieval*. ISMIR 2020. arXiv, Mar. 2020. DOI: 10.48550/arXiv.2003.06226.

[23] Stuart Feldman. "Music affects productivity". In: *Management Review* 80.7 (1991), pp. 6–7.

[24] Evelyn Fix and Joseph Lawson Hodges. "Discriminatory analysis, nonparametric estimation: consistency properties". In: *Report 4, Project no. 21-49* 4 (1951).

[25] Arthur Flexer et al. “Playlist Generation using Start and End Songs.” In: *Proceedings of the 9th International Conference on Music Information Retrieval*. ISMIR 2008. Jan. 2008, pp. 173–178.

[26] Mariagrace Flint. “The Effects of Music on Physical Productivity”. PhD Thesis. Columbus, OH: The Ohio State University, 2010.

[27] JG Fox and ED Embrey. “Music—an aid to productivity”. In: *Applied ergonomics* 3.4 (1972), pp. 202–205.

[28] Adrian Furnham, Sarah Trew, and Ian Sneade. “The distracting effects of vocal and instrumental music on the cognitive test performance of introverts and extraverts”. In: *Personality and Individual Differences* 27.2 (1999), pp. 381–392. ISSN: 0191-8869. DOI: 10.1016/S0191-8869(98)00249-9.

[29] Darryl Griffiths, Stuart Cunningham, and Jonathan Weinel. “A Discussion of Musical Features for Automatic Music Playlist Generation Using Affective Technologies”. In: *Proceedings of the 8th Audio Mostly Conference*. AM 2013. Piteå, Sweden: Association for Computing Machinery, 2013. ISBN: 9781450326599. DOI: 10.1145/2544114.2544128.

[30] W. Bas de Haas, Frans Wiering, and Remco C. Veltkamp. “A geometrical distance measure for determining the similarity of musical harmony”. In: *International Journal of Multimedia Information Retrieval* 2.3 (Sept. 2013), pp. 189–202. ISSN: 2192-662X. DOI: 10.1007/s13735-013-0036-6.

[31] Matthew Hoffman, David Blei, and Perry Cook. “Content-Based Musical Similarity Computation using the Hierarchical Dirichlet Process.” In: *Proceedings of the 9th International Conference of Music Information Retrieval*. ISMIR 2008. Jan. 2008, pp. 349–354.

[32] Paul Jaccard. “The Distribution of the Flora in the Alpine Zone.1”. In: *New Phytologist* 11.2 (1912), pp. 37–50. ISSN: 1469-8137. DOI: 10.1111/j.1469-8137.1912.tb05611.x.

[33] Kurt Jacobson et al. “Music Personalization at Spotify”. In: *Proceedings of the 10th ACM Conference on Recommender Systems*. RecSys 2016. Boston, Massachusetts, USA: Association for Computing Machinery, 2016, p. 373. ISBN: 9781450340359. DOI: 10.1145/2959100.2959120.

[34] Ronald T Kellogg. *Writing Habits and Productivity in Technical Writing*. Tech. rep. US Department of Education, 1982.

[35] Hyoung-Gook Kim, Gee Yeun Kim, and Jin Young Kim. “Music Recommendation System Using Human Activity Recognition From Accelerometer Data”. In: *Transactions on Consumer Electronics* 65.3 (Aug. 2019), pp. 349–358. ISSN: 0098-3063. DOI: 10.1109/TCE.2019.2924177.

[36] Dimitrios Kollias and Stefanos Zafeiriou. *Aff-Wild2: Extending the Aff-Wild Database for Affect Recognition*. 2019. DOI: 10.48550/arXiv.1811.07770.

[37] Paul Lemare. *Boil The Frog*. Jan. 2013. URL: https://musicmachinery.com/2013/01/02/boil-the-frog-2/.

[38] Teresa Lesiuk. "The effect of music listening on work performance". In: *Psychology of music* 33.2 (2005), pp. 173–191.

[39] Elad Liebman, Maytal Saar-Tsechansky, and Peter Stone. "DJ-MC: A Reinforcement-Learning Agent for Music Playlist Recommendation". In: *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*. AAMAS 2015. Istanbul, Turkey: International Foundation for Autonomous Agents and Multiagent Systems, 2015, pp. 591–599. DOI: 10.48550/arXiv.1401.1880.

[40] Hao Liu, Jun Hu, and Matthias Rauterberg. "Music Playlist Recommendation Based on User Heartbeat and Music Preference". In: *Proceedings of the 2009 International Conference on Computer Technology and Development*. Vol. 1. ICCTD 2009. Jan. 2009, pp. 545–549. ISBN: 978-0-7695-3892-1. DOI: 10.1109/ICCTD.2009.246.

[41] Hugo Liu and Push Singh. "ConceptNet—a practical commonsense reasoning tool-kit". In: *BT Technology Journal* 22.4 (2004), pp. 211–226. DOI: 10.1023/B:BTTJ.0000047600.45421.6d.

[42] Malte Ludewig et al. "Effective Nearest-Neighbor Music Recommendations". In: *Proceedings of the ACM Recommender Systems Challenge 2018*. RecSys Challenge 2018. New York, NY, USA: Association for Computing Machinery, Oct. 2018, pp. 1–6. DOI: 10.1145/3267471.3267474.

[43] R. B. Marimont and M. B. Sharpiro. "Nearest Neighbour Searches and the Curse of Dimensionality". In: *IMA Journal of Applied Mathematics* 24.1 (Aug. 1979), pp. 59–70. ISSN: 0272-4960. DOI: 10.1093/imamat/24.1.59.

[44] Hanna Mayer-Benarous et al. "Music Therapy for Children With Autistic Spectrum Disorder and/or Other Neurodevelopmental Disorders: A Systematic Review". In: *Frontiers in Psychiatry* 12 (2021). ISSN: 1664-0640. DOI: 10.3389/fpsyt.2021.643234.

[45] Owen Craigie Meyers. "A mood-based music classification and exploration system". Thesis. Massachusetts Institute of Technology, 2007. URL: https://dspace.mit.edu/handle/1721.1/39337.

[46] Vincenzo Moscato, Antonio Picariello, and Giancarlo Sperlí. "An Emotional Recommender System for Music". In: *IEEE Intelligent Systems* 36.5 (Sept. 2021), pp. 57–68. ISSN: 1941-1294. DOI: 10.1109/MIS.2020.3026000.

[47] Karthik Subramanian Nathan, Manasi Arun, and Megala S Kannan. "EMOSIC — An emotion based music player for Android". In: *Proceedings of the 17th IEEE International Symposium on Signal Processing and Information Technology*. ISSPIT 2017. Dec. 2017, pp. 371–276. DOI: 10.1109/ISSPIT.2017.8388671.

[48] Richard I Newman Jr, Donald L Hunt, and Fen Rhodes. "Effects of music on employee attitude and productivity in a skateboard factory." In: *Journal of Applied Psychology* 50.6 (1966), pp. 493–496. DOI: 10.1037/h0024046.

[49] Elias Pampalk, Tim Pohle, and Gerhard Widmer. "Dynamic Playlist Generation Based on Skipping Behavior". In: *Proceedings of the 6th International*

Conference on Music Information Retrieval. ISMIR 2005. Jan. 2005, pp. 634–637.

[50] Renato Panda et al. "How Does the Spotify API Compare to the Music Emotion Recognition State-of-the-Art?" In: *Proceedings of the 18th Sound and Music Computing Conference*. SMC 2021. Axea sas/SMC Network, July 2021, pp. 238–245. DOI: 10.5281/zenodo.5045100.

[51] Tsang-Long Pao et al. "Comparison between Weighted D-KNN and Other Classifiers for Music Emotion Recognition". In: *Proceedings of the 3rd International Conference on Innovative Computing Information and Control*. ICICIC 2008. June 2008, pp. 530–530. DOI: 10.1109/ICICIC.2008.679.

[52] Sungjoon Park et al. "Dimensional Emotion Detection from Categorical Emotion". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 4367–4380. DOI: 10.18653/v1/2021.emnlp-main.358.

[53] Steffen Pauws, Wim Verhaegh, and Mark Vossen. "Fast Generation of Optimal Music Playlists using Local Search". In: *Proceedings of the 7th International Conference on Music Information Retrieval*. ISMIR 2006. 2006.

[54] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[55] Martin Pichl, Eva Zangerle, and Günther Specht. "Combining Spotify and Twitter Data for Generating a Recent and Public Dataset for Music Recommendation". In: *Proceedings of the 26th Workshop Grundlagen von Datenbanken*. GvDB 2014, pp. 35–40.

[56] Anthony T. Pinter et al. "P4KxSpotify: A Dataset of Pitchfork Music Reviews and Spotify Musical Features". In: *Proceedings of the International AAAI Conference on Web and Social Media* 14 (May 2020), pp. 895–902. ISSN: 2334-0770. DOI: 10.1609/icwsm.v14i1.7355.

[57] Robert Plutchik. "Chapter 1 - A General Pscyhoevolutionary Theory of Emotion". In: *Theories of Emotion*. Ed. by Robert Plutchik and Henry Kellerman. Academic Press, Jan. 1980, pp. 3–33. ISBN: 978-0-12-558701-3. DOI: 10.1016/B978-0-12-558701-3.50007-7.

[58] Wolfram Research. *Cosine Distance*. 2007. URL: https://reference.wolfram.com/language/ref/CosineDistance.html.

[59] Gordon Reynolds et al. "Towards a personal automatic music playlist generation algorithm: the need for contextual information". In: (2007), pp. 84–89.

[60] Paul Rosen and Ghulam Jilani Quadri. *LineSmooth: An Analytical Framework for Evaluating the Effectiveness of Smoothing Techniques on Line Charts*. Vol. 27. 2. 2021, pp. 1536–1546. DOI: 10.1109/TVCG.2020.3030421.

[61] Mikhail Rumiantcev and Oleksiy Khriyenko. "Emotion Based Music Recommendation System". In: FRUCT 26. 2020, pp. 639–645.

[62] James A Russell. "A circumplex model of affect." In: *Journal of personality and social psychology* 39.6 (1980), p. 1161.

[63] Keigo Sakurai et al. "Music Playlist Generation Based on Graph Exploration Using Reinforcement Learning". In: *Proceedings of the IEEE 3rd Global Conference on Life Sciences and Technologies*. LifeTech 2021. Mar. 2021, pp. 53–54. DOI: 10.1109/LifeTech52111.2021.9391870.

[64] Basna Mohammed Salih Hasan and Adnan Mohsin Abdulazeez. "A Review of Principal Component Analysis Algorithm for Dimensionality Reduction". In: *Journal of Soft Computing and Data Mining* 2.1 (Apr. 2021), pp. 20–30.

[65] Arnaja Sen et al. "Music Playlist Generation using Facial Expression Analysis and Task Extraction". In: *Annales Universitatis Mariae Curie-Sklodowska, sectio AI – Informatica* 16.2 (Dec. 2017), pp. 1–6. ISSN: 2083-3628. DOI: 10.17951/ai.2016.16.2.1.

[66] Jangid Sheetal Kailash et al. "Behavioural, Emotional State Based Music Selection & Playlist Generating Player". In: *International Journal Of Current Engineering And Scientific Research*. IJCESR 4.12 (2017), pp. 39–43. DOI: 10.21276/ijcesr.

[67] Federico Simonetta et al. "Symbolic Music Similarity through a Graph-Based Representation". In: *Proceedings of the Audio Mostly 2018 on Sound in Immersion and Emotion*. AM 2018. New York, NY, USA: Association for Computing Machinery, Sept. 2018, pp. 1–7. ISBN: 978-1-4503-6609-0. DOI: 10.1145/3243274.3243301.

[68] Archana Singh, Avantika Yadav, and Ajay Rana. "K-means with Three different Distance Metrics". In: *International Journal of Computer Applications* 67.10 (2013).

[69] Spotify. *Web API — Spotify for Developers*. URL: https://developer.spotify.com/documentation/web-api.

[70] Carl Thomé, Sebastian Piwell, and Oscar Utterbäck. *Musical Audio Similarity with Self-supervised Convolutional Neural Networks*. Feb. 2022. DOI: 10.48550/arXiv.2202.02112. (Visited on 03/28/2023).

[71] Yuchi Tian et al. "DeepTest: Automated Testing of Deep-Neural-Network-Driven Autonomous Cars". In: *Proceedings of the 40th International Conference on Software Engineering*. ICSE 2018. Gothenburg, Sweden: Association for Computing Machinery, 2018, pp. 303–314. ISBN: 9781450356381. DOI: 10.1145/3180155.3180220.

[72] Petri Toiviainen, Mauri Kaipainen, and Jukka Louhivuori. "Musical timbre: Similarity ratings correlate with computational feature space distances". In: *Journal of New Music Research* 24.3 (Sept. 1995), pp. 282–298. ISSN: 0929-8215. DOI: 10.1080/09298219508570686.

[73] Kevin N White. "The Effects of Background Music in the Classroom on the Productivity, Motivation, and Behavior of Fourth Grade Students." MA thesis. Columbia, SC: Columbia College, 2007.

[74] In-Kwon Yeo and Richard A. Johnson. “A New Family of Power Transformations to Improve Normality or Symmetry”. In: *Biometrika* 87.4 (2000), pp. 954–959. ISSN: 00063444.

[75] Hamed Zamani et al. *An Analysis of Approaches Taken in the ACM RecSys Challenge 2018 for Automatic Music Playlist Continuation*. Aug. 2019.

[76] Kejun Zhang et al. “The PMEmo Dataset for Music Emotion Recognition”. In: *Proceedings of the 2018 ACM International Conference on Multimedia Retrieval*. ICMR 2018. Yokohama, Japan: Association for Computing Machinery, 2018, pp. 135–142. ISBN: 9781450350464. DOI: 10.1145/3206025.3206037.