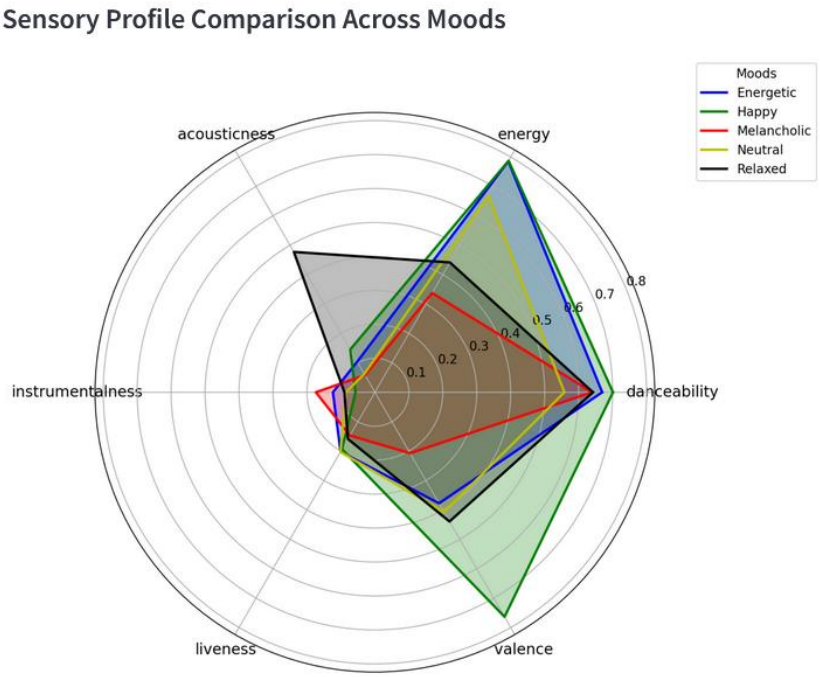
Final Project

Machine Learning and the Music Industry:

Tuning Algorithms for Listening Ears



SimplyFy Recommender Web Application

Presented By

Gabriel Ferreira, Pedro Barros

Table of Contents

Introduction

Background

Spotify is one of the market pioneers in music, podcast, and audiobook streaming as a service. Since its release in 2008, Spotify has amassed over 640 million users, 252 million subscribers(Spotify, n.d.) and a market cap estimated to be at over $90 billion USD as of 11/16/2024, according to Yahoo Finance(Yahoo Finance, n.d.). This puts Spotify in a market leader position, comfortably holding the largest market share (30.5%), nearly doubling the share of any other players in the segment(Ng, 2024). Despite its success, entering phrases like “Spotify recommendations” in any search engine will return plenty of user-built alternatives and complaints about the platform’s recommendation algorithms. The MIT Technology Review article “How to break free of Spotify’s algorithm” highlights the frustrations experienced by Spotify users, including the disappointment caused by recommendation algorithms that underdeliver. The complex algorithms bucketize all the information collected from most popular searches, users’ listening habits, and popular music into three main categories: everyday listening, sound-alikes, and everything else; the first one is the one tapped into most often, with the third one being the one that is least utilized by the algorithm. This causes the algorithm to repeatedly gravitate towards the familiar, creating an endless cycle of déjà vu to listeners, who seldom discover new music that aligns with their preferences(Ng, 2024). The objective of this project is not to reinvent Spotify’s recommendation algorithm, but rather to adopt an "Occam’s Razor" approach by simplifying the process through elimination of assumptions about what the user would like to enjoy. Instead, the new approach seeks to engage users directly, asking them to identify tracks that resonate with their tastes and, based on their input, recommending similar tracks that closely match what they are truly seeking.

Business Problem and Current Approach

The position as a market leader does not come free, especially in the streaming sector. The ease with which users can terminate and start different services at any given time creates inherent volatility in the market that can quickly change the landscape and, as a consequence, its leaders. User satisfaction comes as a cornerstone of this industry; a satisfied user is one that is less likely to exercise this freedom of choice and leave its current service for a new one.

With the ever-growing music catalogs available to users to enjoy and with the recent advancements in machine learning and artificial intelligence used by all competitors in the segment, a tailored listening experience for each user has shifted from a nice-to-have into another deciding factor when choosing what platform to adhere. Spotify in particular attempts to improve user experience by utilizing a hybrid approach to music recommendation. The platform uses a multifaceted approach that involves applying collaborative filtering, content-based analysis, social analysis, natural language processing (NLP) and many other methodologies; the hybrid approach attempts to cover different aspects of the listening experience, while mitigating possible drawbacks of the individual methodologies and creating an ever-evolving profile of its users based on their listening history, content preferences, and interactions with the platform to best deliver Spotify's catalog(Chen, 2024). Other studies proposing an alternative to Spotify's recommendation system often follow the same route, increasing the complexity in the recommendation system by adding more models, algorithms, and variables to an already convoluted approach; incorporating album art color mapping, lyric-based sentiment analysis through NLP and sentiment lexicon models (e.g.: VADER) (Jr et al., 2021) are some of the strategies employed as an alternative (or addition) to Spotify's recommender system. While the added complexity may prove fruitful in some cases, it may cause an already convoluted system, like Spotify's recommendation system, to fall into the Information Overload Paradox, creating difficulties for algorithms to distinguish noise over signal and yield undesired outputs by users.

Common issues faced by Spotify users include dissatisfaction with initial recommendations (cold-start problem) and failure to adapt to users' listening habits that can rapidly change(Schedl et al., 2018). On the other hand, some platforms like Deezer attempt a more user-centric approach with an initial data collection process to guide its algorithms, which may create a slightly more pleasant overall user experience(De Assunção & Zaina, 2022). These issues further highlight the importance of understanding and valuing what the user wants (and expects) out of interacting with music streaming applications in a simple, straightforward manner. After all, the user is the one with the power of choice in a saturated market with plenty of solutions to fulfill its listening needs, such as receiving meaningful, appropriate music recommendations.

Proposed Approach to Business Problem

The hybrid approach adopted by Spotify tries (and succeeds) to solve many problems but appears to often drift from focusing on the individual user in favor of collaborative filtering and other social-

based algorithms that generally address the community of users instead of the individual user. This can cause dissonance between what the individual user wants or expects and what the algorithm recommends. The proposed solution seeks to bridge that gap by offering the user a direct channel to communicate what is desired and expected from the recommendations, which will be used to tailor recommendations for the user. This approach makes the user and the object of their current interest (i.e.: a track, the idea of a track, or an emotion) an integral part of the recommendation process, without including complexities such as popular tracks at the time, the users' history, or what the community of users might have liked into that process, which might not necessarily represent any added value to the users' experience at that time. Not only is this expected to reduce user frustration due to failed expectations, but it would significantly reduce the complexity of Spotify's recommendation process, which would also free resources to be allocated in more important tasks or with other users that may want a more complex recommendation experience where all other variables are considered. The solution will be made available to users via web application.

Data Source

The data source utilized in this study is a comma-separated values file with over 30,000 tracks collected from the Spotify API utilizing the *spotifyr* library (for R language) and made available via Kaggle. The dataset contains track characteristics (e.g.: acousticness, instrumentalness, valence, tempo, etc.) that were extracted utilizing Spotify's proprietary algorithms alongside identifying features of tracks (e.g.: unique id, song name, album, artist, etc.). The data dictionary for the data source used is as follows:

variable	class	description
track_id	character	Song unique ID
track_name	character	Song Name
track_artist	character	Song Artist
track_popularity	double	Song Popularity (0-100) where higher is better
track_album_id	character	Album unique ID
track_album_name	character	Song album name
track_album_release_date	character	Date when album released
playlist_name	character	Name of playlist

playlist_id	character	Playlist ID
playlist_genre	character	Playlist genre
playlist_subgenre	character	Playlist subgenre
danceability	double	Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable.
energy	double	Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. For example, death metal has high energy, while a Bach prelude scores low on the scale. Perceptual features contributing to this attribute include dynamic range, perceived loudness, timbre, onset rate, and general entropy.
key	double	The estimated overall key of the track. Integers map to pitches using standard Pitch Class notation . E.g. 0 = C, 1 = C♯/D♭, 2 = D, and so on. If no key was detected, the value is -1.
loudness	double	The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing relative loudness of tracks. Loudness is the quality of a sound that is the primary psychological correlate of physical strength (amplitude). Values typical range between -60 and 0 db.
mode	double	Mode indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by 1 and minor is 0.

speechiness	double	Speechiness detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value. Values above 0.66 describe tracks that are probably made entirely of spoken words. Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either in sections or layered, including such cases as rap music. Values below 0.33 most likely represent music and other non-speech-like tracks.
acousticness	double	A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.
instrumentalness	double	Predicts whether a track contains no vocals. "Ooh" and "aah" sounds are treated as instrumental in this context. Rap or spoken word tracks are clearly "vocal". The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content. Values above 0.5 are intended to represent instrumental tracks, but confidence is higher as the value approaches 1.0.
liveness	double	Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. A value above 0.8 provides strong likelihood that the track is live.
valence	double	A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry).

tempo	double	The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece and derives directly from the average beat duration.
duration_ms	double	Duration of song in milliseconds

Limitations

The most notable limitation presented by the proposed approach is mostly related to the restrictions imposed by utilizing an offline data source. Not only does Spotify's API have restrictions regarding maximum requests allowed, which would make collecting enough track data for a meaningful track catalog to be used with the proposed algorithm laborious and time-consuming. Utilizing an offline data source limits the recommendation options and user track selection, as only the track contained in the dataset are possible candidates to be elected by the user or recommended, which may hinder the solution ineffective for some users.

Another important limitation to be discussed is the impossibility to quantify the relevance to the tracks recommended by the algorithm to the user. Because the algorithm proposed objectively groups and recommends tracks based on similarity metrics, the relevance of the tracks recommended and, ultimately the performance of the algorithm in fulfilling its purpose of recommending tracks to user, can only be objectively tested and validated based on the subjective opinion of users regarding the recommendations. In other words, the performance of the algorithm is intimately linked to the perception of users regarding the recommendations proposed by the algorithm, which require alternative methods of quantifying its performance from a user-centric lens that may still be subjective and hard to explore(Schedl et al., 2018). Some possible ways to measure the performance of the recommendations that were not conducted as part of this study could include surveys, interviews, and behavioral data analysis to better understand how users interact with Spotify before and after the algorithm proposed in this study.

Lastly, the algorithm's performance may suffer with the limited context which expectedly might come as a result of complexity reduction, the foundational element of the proposed solution. The suggested mitigating mechanisms to minimize the drawbacks of adopting this complexity reduction mentality was to utilize clustering algorithms to introduce some context to the tracks that are candidates to be recommended; each cluster could offer more context (e.g.: similar year of release,

similar genres, etc.) by grouping similar tracks before applying similarity metrics and other devices to recommend tracks, which could yield more contextualized recommendations.

Exploratory Data Analysis & Data Preprocessing

Identifying and Treating Null Values

Only 5 instances of null values were found in the dataset. Since removing those nulls present no integrity concerns for the analysis or future processes involving the dataset, this will be the method of choice to treat null values.

```
track_id                  0
track_name                5
track_artist              5
track_popularity          0
track_album_id            0
track_album_name          5
track_album_release_date  0
playlist_name             0
playlist_id               0
playlist_genre            0
playlist_subgenre         0
danceability              0
energy                    0
key                       0
loudness                  0
mode                      0
speechiness               0
acousticness              0
instrumentalness          0
liveness                  0
valence                   0
tempo                     0
duration_ms               0
dtype: int64
```

Identifying and Treating Duplicates

The initial idea was to use the unique identifier **track_id** to identify and treat duplicated songs. However, some tracks were still duplicated after this approach was carried out. A common reason for this phenomenon to happen stems from the fact that certain songs may have been part of more than one album or may have been remastered more than once. In all instances, duplicated songs had the same main characteristics (e.g.: acousticness, speechiness, loudness, etc.), deeming the identification of which instance should be removed from the dataset irrelevant, as either instance would present the same characteristics. Further investigation pointed to another source of

duplicates; since a song can be present in more than one playlist, they can appear more than once in the dataset. Considering that the study's focus is on individual tracks, their characteristics, and recommendations to listeners, all playlist information and duplicated tracks will be removed from the dataset. The variables utilized to identify duplicates were expanded to include **track_name, track_artist**, which ensures that only unique combinations of these variables are maintained:

```
songs.drop(columns=['playlist_name', 'playlist_id','playlist_genre', 'playlist_subgenre'], inplace=True)

rows,columns = songs.shape

print(f'Rows: {rows:,}\nColumns: {columns}')
```

```
Rows: 32,828
Columns: 19
```

```
songs.drop_duplicates(subset=['track_name','track_artist'], inplace=True)

rows,columns = songs.shape

print(f'Rows: {rows:,}\nColumns: {columns}')
```

```
Rows: 26,229
Columns: 19
```

Identifying and Treating Dtypes

The only feature whose dtype was not optimally represented in the dataset was the **track_album_release_date** feature. Originally, this feature was an *object* type, which not only does not align with the contents of the variable but also hinders the capability to extract date elements (e.g.: year) from the variable. The best option to better align the feature with its contents and to allow for easier manipulation and element extraction from the variable would be to convert the **track_album_release_date** from the *object* type to the *datetime* type. However, some instances only have the year available, which raises errors when trying to convert the variable into the *datetime* dtype. For that reason, a function was created to perform the following:

- If there is no date available (null value), the observation is not formatted at all. This is done so null values would still be visible and can be treated appropriately, if they exist.
- If the date field in an observation only has 4 characters, only the year of release is available. With that, the first calendar day and month for that year were imputed and the date field will be formatted as yyyy-01-01.

- If the date field in an observation only has 7 characters, only the year and month of release are available. With that, the first calendar day for that year and month combination was imputed and the date field will be formatted as yyyy-mm-01.
- Otherwise, the date field in an observation is considered to be complete and no other formatting is necessary.

After formatting the date field, each observation is appended to a list with standardized date formats, which is then converted to the *datetime* and returned by the function. That way, the desired date information can be properly extracted and manipulated as needed in future processes.

```python
def standardize_date(dates):
    """
    Standardizes a list of date strings to the format 'YYYY-MM-DD'.

    Parameters:
        dates (iterable): An iterable containing date strings in various formats
                          ('YYYY', 'YYYY-MM', 'YYYY-MM-DD').

    Returns:
        pd.Series: A Pandas Series with dates converted to datetime format, where:
                   - 'YYYY' is converted to 'YYYY-01-01'
                   - 'YYYY-MM' is converted to 'YYYY-MM-01'
                   - 'YYYY-MM-DD' remains unchanged
                   Invalid dates will be set as NaT (Not a Time).
    """
    standardized_dates = []
    for date in dates:
        if pd.isna(date):
            standardized_dates.append(date)
        elif len(date) == 4:
            standardized_dates.append(f"{date}-01-01")
        elif len(date) == 7:
            standardized_dates.append(f"{date}-01")
        else:
            standardized_dates.append(date)

    return pd.to_datetime(standardized_dates, errors='coerce')


songs['track_album_release_date'] = standardize_date(songs['track_album_release_date'])
songs['release_year'] = songs['track_album_release_date'].dt.year
songs = songs.drop(columns=['track_album_release_date'])
songs.head()
```

Identifying and Treating Outliers

The method of choice to identify and classify possible outliers in the dataset was the Tukey's Fence method, where IQR represents the interquartile range and:

- Lower bound outliers are classified as values < Q1 - 1.5*IQR, and
- Upper bound outliers are classified as values > Q3 + 1.5*IQR.

Following the methodology, the following number of instances were flagged as possible outliers:

```
Lower Bound Outliers:

track_popularity       0
danceability         254
energy               214
key                    0
loudness             772
mode                   0
speechiness            0
acousticness           0
instrumentalness       0
liveness               0
valence                0
tempo                  5
duration_ms          115
release_year        2335
dtype: int64

Total Lower Bound Outliers in Dataframe:

3695
```

```
Upper Bound Outliers:

track_popularity       0
danceability           0
energy                 0
key                    0
loudness               2
mode                   0
speechiness         2467
acousticness        1718
instrumentalness    5667
liveness            1527
valence                0
tempo                430
duration_ms         1017
release_year           0
dtype: int64

Total Upper Bound Outliers in Dataframe:

12828
```

Given that most features in this dataset numerically represent song characteristics and are bound to a scale (e.g., 0 to 1, 0 to 100, etc.), they semantically function as ordinal indicators rather than true quantitative measures. For instance, a danceability score of 0.80 does not imply the track is twice as danceable as one with a score of 0.40; instead, it simply ranks higher in confidence about the track's danceability. Similarly, a track with an acousticness score of 0.50 has stronger acoustic qualities than one with a score of 0.35, but this difference represents a qualitative rank rather than a strict quantitative ratio. For that reason, the only features that may be considered to have outliers that may require treatment, as defined by extreme values significantly dissonant from the remaining observations, are:

- **duration_ms**
- **tempo**
- **release_year**

Given that **tempo** and **release_year** relay important information about track characteristics that certainly impact public perception and track categorization (e.g.: songs with fast tempo from the 1980s may be different than songs with fast tempo from the 1960s), they will be preserved as-is to maintain the original characteristics of the songs and what they represent to listeners.
The **duration_ms** will require further investigation and study to determine if it requires treatment and how to treat it.

Treating the *duration_ms* Feature

Upper Bound Outliers

A visual inspection of the top 25 songs by **duration_ms** was carried out. The analysis of songs with notable upper bound outliers in the **duration_ms** feature reveals no immediate concerns. Many well-known tracks, like Kashmir and American Pie, and popular bands, like Led Zeppelin and The Who, appear among these entries, cementing that these outliers are genuine instances of songs with longer durations. Therefore, no treatment is required for these outliers, as they accurately represent real, extended-length tracks.

Lower Bound Outliers

A similar process as the one carried out for upper bound outlier was carried out, and a visual inspection of the bottom 25 songs by **duration_ms** was carried out. The visual inspection of songs with notable lower bounds outliers in the **duration_ms** feature reveals that many of the tracks appear to be interlude and transition tracks. These tracks are often instrumental pieces used to bridge sections of an album or presentation and help in the storytelling and atmosphere of the pieces. For that reason, they are not necessarily meant to be enjoyed by themselves but rather as part of a greater piece. Since these tracks are so dissonant from the rest of the tracks in style and utility, the lower bound of the **duration_ms** feature will be limited to instances above the 1st percentile, and any instance that presents a duration below that threshold will be removed. A last visual inspection of the bottom 25 songs by **duration_ms** remaining after treating the lower bound outliers reveal that the instances accurately represent real, shortened-length tracks, deeming the lower bound outlier treatment successful.

Variable Encoding

The **track_artist_label** and **track_album_id_label** were label encoded via LabelEncoder, so the numerical representation of their contents could be used in future processes:

```
from sklearn.preprocessing import LabelEncoder

# Create a LabelEncoder
encoder = LabelEncoder()
songs['track_artist_label'] = encoder.fit_transform(songs['track_artist'])
songs['track_album_id_label'] = encoder.fit_transform(songs['track_album_id'])
songs.head()
```

Working Dataset

After the Initial data cleaning and preprocessing steps took place, the following dataset will be used going forward in this study:

```
The following variables represent the working variables for this study:
track_id
track_name
track_artist
track_popularity
track_album_id
track_album_name
danceability
energy
key
loudness
mode
speechiness
acousticness
instrumentalness
liveness
valence
tempo
duration_ms
release_year
track_artist_label
track_album_id_label


Rows: 25,966
Columns: 21
```

Methodology & Results

Recommendation Algorithm - Track Input

The general idea of this alternative for Spotify's recommendation algorithm was to utilize similarity metrics that were commonly applied in text similarity problems throughout the MIS-5460 course and other researched applications (e.g.: Bray-Curtis Dissimilarity for ecological counts) and expand their use to compare vectors representing track characteristics (e.g.: speechiness, danceability, energy etc.) instead. The similarity scores between tracks are captured, sorted, and, finally, returned as recommendations for users that wish to listen to similar tracks to the one initially input in the beginning of the process.

A possible challenge faced with this approach could stem from the lack of context in the recommendations (e.g.: user history). To mitigate that possible challenge, clustering algorithms will be used in an attempt to restore some of the contextual clues for the recommendations. The clustering algorithm will consider the characteristics of tracks and group them in a meaningful way to isolate comparisons amongst tracks that fit the same context and general characteristics better. An example would be a user that would like to have recommendations for songs similar to Led Zeppelin's Immigrant Song, a powerful, high-energy rock anthem that epitomizes the band's

electrifying sound from the 1970s with primal vocals, generating an exhilarating atmosphere. That user would likely want to be recommended other tracks that stylistically resemble the sound of that track and that specific hard rock band from that period, which may present characteristics (e.g.: loudness, energy, key, etc.) that closely align with unrelated genres and periods, such as 2000's pop tracks, for example. This would generate disconnected recommendations (e.g.: Shakira songs to a Led Zeppelin fan), causing user frustration. The clustering algorithm would likely group the Led Zeppelin tracks with other tracks from the 1970's that resemble the features of Led Zeppelin more closely, which would add essential context regarding the period (and other characteristics) to differentiate both genres and eras.

Clustering Process

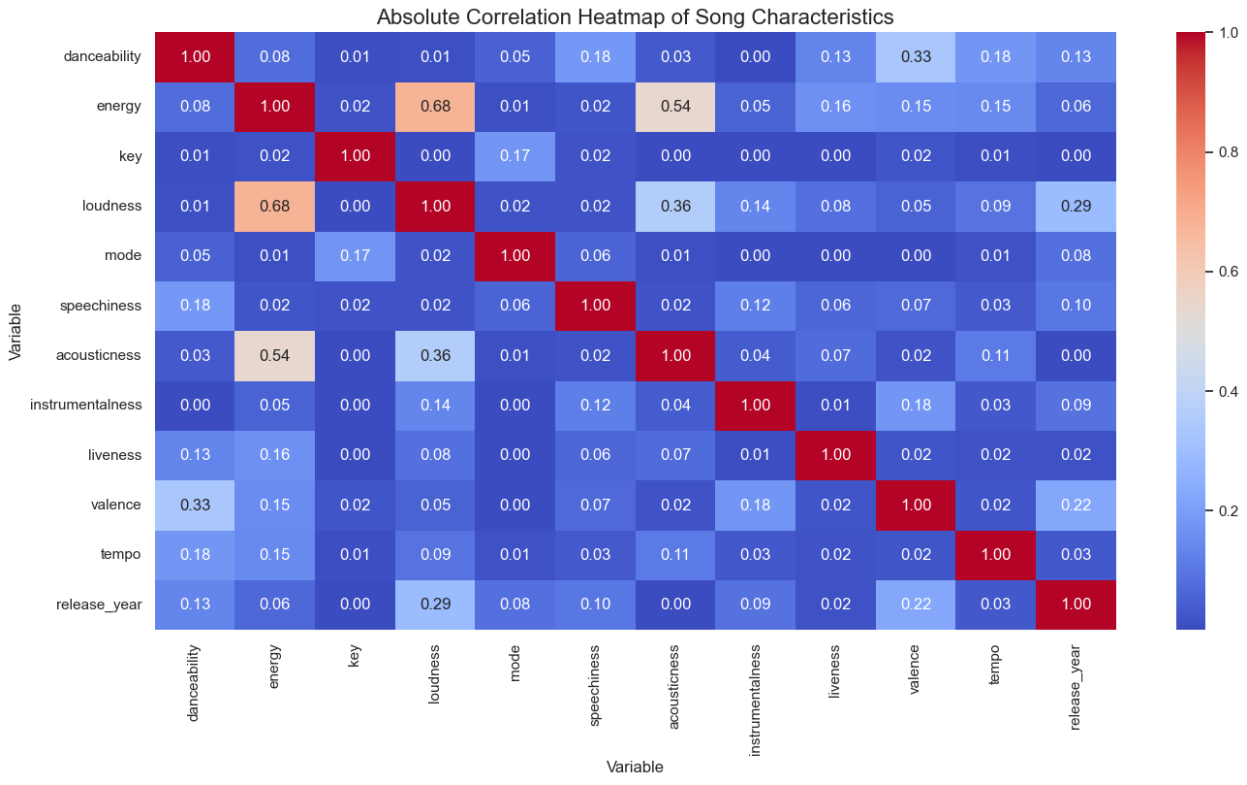
Three clustering algorithms were identified as possible candidates to perform that task well; the algorithms were picked based on diversity in clustering methodology (hierarchical, model-based, and partitional), each presenting inherent different advantages and disadvantages of their own:

1. **Agglomerative Clustering - Hierarchical Clustering**
    - Flexible, as many distance/linkage measures can be used.
    - Works well for non-convex clusters.
    - Sensitive to noise and outliers.

2. **Gaussian Mixture Model (GMM) - Model-based Clustering**
    - Uses a probabilistic approach to assign clusters.
    - Handles overlapping clusters well due to the probabilistic approach (i.e.: small differences may yield differences in probability to assign clusters).
    - Assumes Gaussian distributions, which may not reflect this use case.

3. **K-Means Clustering - Partitional Clustering**
    - Computationally efficient and easy to implement.
    - Flexible and adaptable to many scenarios (e.g.: image compression, species identification, etc.).
    - Sensitive to noise and outliers.

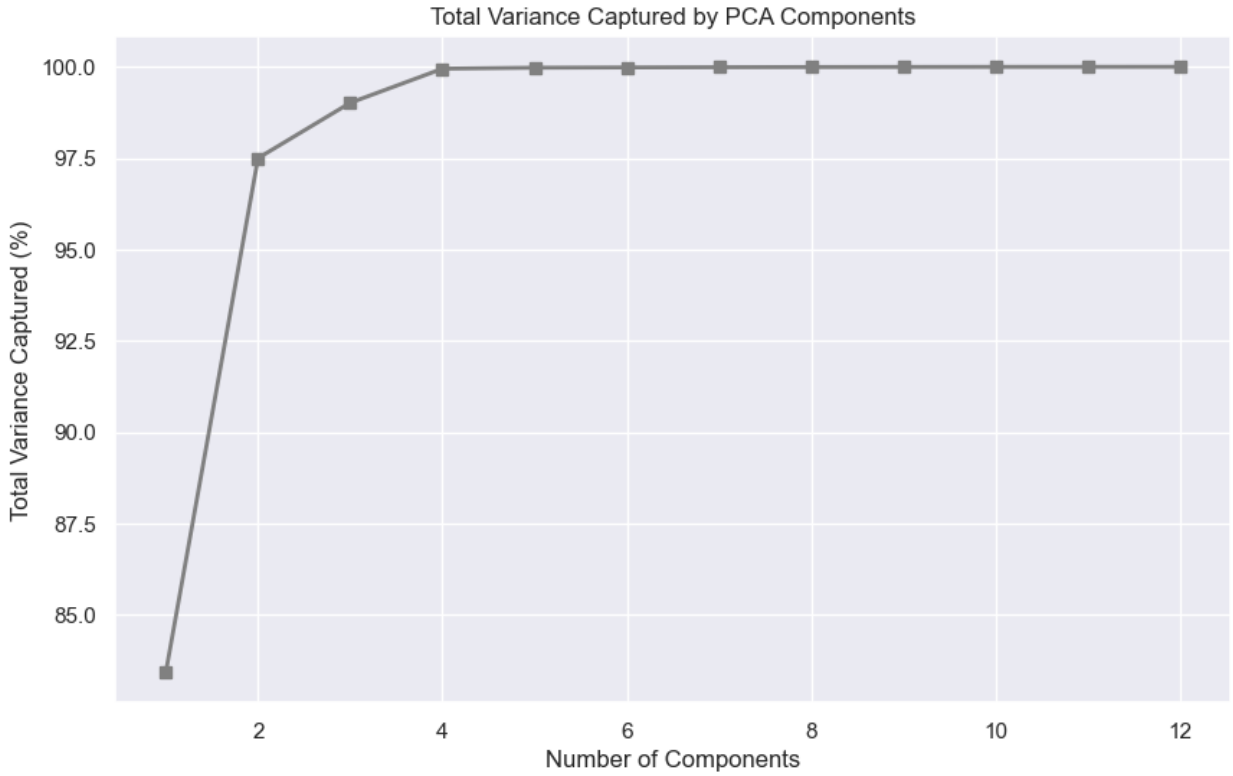
Since the focus of clustering is to identify patterns and shared context among songs and their characteristics, features that do not express characteristics inherent to the tracks will not be used as a basis for recommendation. *Track_popularity,* for example, is an external characteristic of the track (i.e.: not descriptive of the track itself or how it sounds). Including that variable may cause popular tracks to be clustered (and recommended) together, which may hurt music democratization by alienating less popular tracks. This has been a topic of discussion regarding Spotify's recommender system, as popularity influencing the algorithm may cause artists that depend on the platform to change their creative mindset to adapt to what's popularly recommend, may restrict musical discovery by users that only receive popular recommendations, and may perpetuate the standardization of music by corporations that need to sell their (recommended) product(Hodgson, 2021). For those reasons, that variable will be excluded from the algorithm. Aside from that, the *track_artist_label* and *track_album_id_label* features were causing clusters to revolve around those items and be clustered ordinally (e.g.: cluster 0 has *track_artist_label* from *0* to *x*, cluster 1 from *x* to *y*, etc.), which does not truly reflect a pattern or tendency, so they will also be removed going forward in the study.



Some of the features studied present strong enough correlation to suspect that the dataset may present multicollinearity issues. The absolute correlation heatmap below shows that loudness and

energy and acousticness and energy are good candidates in which multicollinearity issues may be present, which may result in poor performance for clustering algorithms. Furthermore, some of the features may be semantically similar, thus offering similar context to the clustering algorithm or may not offer any additional valuable information, resulting in increased complexity and reduced performance of the clustering algorithms. For that reason, PCA will be evaluated as an alternative to:

- Reduce complexity by reducing dimensionality
- Mitigate possible multicollinearity issues
- Mitigate the impact of noise (noise over signal issue)
- Possibly improve clustering performance



Given the marginal explained variance gains when adding more components, two components will be the quantity of PCA components chosen to carry forward into the study. 97.5% of the variance is explained with only two components, which enforce the initial conjuncture that there appears to be redundant features in the dataset (i.e.: offer the same information or are correlated), deeming the data structure conducive to dimensionality reduction. Using PCA is also expected to help the clustering algorithms focus on the components that are truly driving variances, instead of shifting

focus towards noise, correlated, or irrelevant features that bring no additional informational value to the clusters.

Lastly, applying PCA goes together with the initial proposal to reduce the complexity currently found in music recommendation algorithms, which is a pillar of the philosophy supporting this study.

Model Evaluation

The performance of the clustering algorithms will be determined by analyzing the following metrics:

Davies-Bouldin Index (documentation)

- Measures the average similarity ratio of clusters, where similarity is defined as the ratio of intra-cluster distance to inter-cluster distance. Clusters with less dispersion and less similar (more distant from each other) score better. *Low scores are favorable*, indicating compact clusters with little to no overlapping.
- Focuses on compactness and separation.

Calinski-Harabasz Score (documentation)

- Measures the ratio of the between-cluster dispersion to the within-cluster dispersion. *High scores are favorable*, indicating distinct and well-separated clusters. Low scores indicate clusters that are dispersed and not well-defined.
- Emphasizes the ratio of intra- and inter-cluster variance.

Silhouette Score (documentation)

- Evaluates how similar each point in a cluster is to points in its own cluster compared to points in the nearest cluster. *High scores are favorable*, indicating that samples are closer to their assigned cluster than to other clusters.
- Balances compactness and separation but is more interpretable for individual samples.

The performance of the models will be tested and analyzed following the metrics discussed with and without PCA reduction applied for different cluster combinations ranging from 2 through 18 in steps of 2. The efficacy of the PCA reduction will be attested (or not) by comparing the metrics between model with and without the technique; the best performance between both scenarios will decide whether the models pre or post PCA will be scrutinized to decide the best performing model.

The following performance results were collected following model testing and evaluation:

	n_clusters	Raw_Davies_Bouldin	PCA_Davies_Bouldin	Raw_Calinski_Harabasz	PCA_Calinski_Harabasz	Raw_Silhouette_Score	PCA_Silhouette_Score
KMeans Clusters with 18 clusters	18	1.03	0.81	21155.0	33913.0	0.28	0.38
KMeans Clusters with 16 clusters	16	1.02	0.78	22410.0	34985.0	0.27	0.41
KMeans Clusters with 14 clusters	14	0.99	0.78	22106.0	34414.0	0.34	0.44
KMeans Clusters with 12 clusters	12	1.02	0.82	23893.0	33814.0	0.29	0.42
KMeans Clusters with 10 clusters	10	0.98	0.82	25492.0	33712.0	0.33	0.42
KMeans Clusters with 8 clusters	8	0.87	0.76	27347.0	34550.0	0.39	0.46
KMeans Clusters with 6 clusters	6	0.97	0.72	25151.0	31721.0	0.39	0.44
KMeans Clusters with 4 clusters	4	0.76	0.70	30716.0	34652.0	0.48	0.52
KMeans Clusters with 2 clusters	2	0.81	0.79	25533.0	26855.0	0.45	0.47
Gaussian Mixture Clusters with 18 clusters	18	18.87	2.78	352.0	4811.0	-0.36	-0.02
Gaussian Mixture Clusters with 16 clusters	16	10.53	3.21	452.0	3861.0	-0.28	-0.12
Gaussian Mixture Clusters with 14 clusters	14	11.79	3.50	513.0	4580.0	-0.30	-0.07
Gaussian Mixture Clusters with 12 clusters	12	10.84	3.04	547.0	3933.0	-0.28	-0.10
Gaussian Mixture Clusters with 10 clusters	10	11.18	2.61	277.0	4557.0	-0.24	-0.06
Gaussian Mixture Clusters with 8 clusters	8	12.82	10.98	288.0	3025.0	-0.22	-0.13
Gaussian Mixture Clusters with 6 clusters	6	7.85	2.01	306.0	8538.0	-0.21	0.04
Gaussian Mixture Clusters with 4 clusters	4	17.29	5.55	275.0	2328.0	-0.15	-0.01
Gaussian Mixture Clusters with 2 clusters	2	17.92	2.59	40.0	2487.0	-0.03	0.18
Agglomerative Clusters with 18 clusters	18	1.08	0.84	18417.0	28732.0	0.24	0.37
Agglomerative Clusters with 16 clusters	16	1.10	0.89	18645.0	28491.0	0.22	0.37
Agglomerative Clusters with 14 clusters	14	1.08	0.89	19197.0	28313.0	0.26	0.35
Agglomerative Clusters with 12 clusters	12	1.01	0.89	20158.0	28129.0	0.27	0.39
Agglomerative Clusters with 10 clusters	10	1.04	0.86	21923.0	28990.0	0.32	0.41
Agglomerative Clusters with 8 clusters	8	0.97	0.81	24522.0	30062.0	0.37	0.42
Agglomerative Clusters with 6 clusters	6	0.92	0.82	24787.0	29351.0	0.35	0.40
Agglomerative Clusters with 4 clusters	4	0.82	0.78	25317.0	31393.0	0.46	0.51
Agglomerative Clusters with 2 clusters	2	0.80	0.76	22140.0	22937.0	0.42	0.44

The performance outcomes in models where PCA applied present significantly better results in comparison to its counterpart where the technique was bypassed. With that, the models with PCA will be further scrutinized to determine the best algorithm and quantity of clusters to be carried forward in the study.

Top 5 Best Performing Models by Calinski-Harabasz Score

	n_clusters	PCA_Davies_Bouldin	PCA_Calinski_Harabasz	PCA_Silhouette_Score
KMeans Clusters with 16 clusters	16	0.78	34985.0	0.41
KMeans Clusters with 4 clusters	4	0.70	34652.0	0.52
KMeans Clusters with 8 clusters	8	0.76	34550.0	0.46
KMeans Clusters with 14 clusters	14	0.78	34414.0	0.44
KMeans Clusters with 18 clusters	18	0.81	33913.0	0.38

Top 5 Best Performing Models by Davies-Bouldin Index

	n_clusters	PCA_Davies_Bouldin	PCA_Calinski_Harabasz	PCA_Silhouette_Score
KMeans Clusters with 4 clusters	4	0.70	34652.0	0.52
KMeans Clusters with 6 clusters	6	0.72	31721.0	0.44
KMeans Clusters with 8 clusters	8	0.76	34550.0	0.46
Agglomerative Clusters with 2 clusters	2	0.76	22937.0	0.44
KMeans Clusters with 16 clusters	16	0.78	34985.0	0.41

Top 5 Best Performing Models by Silhouette Score

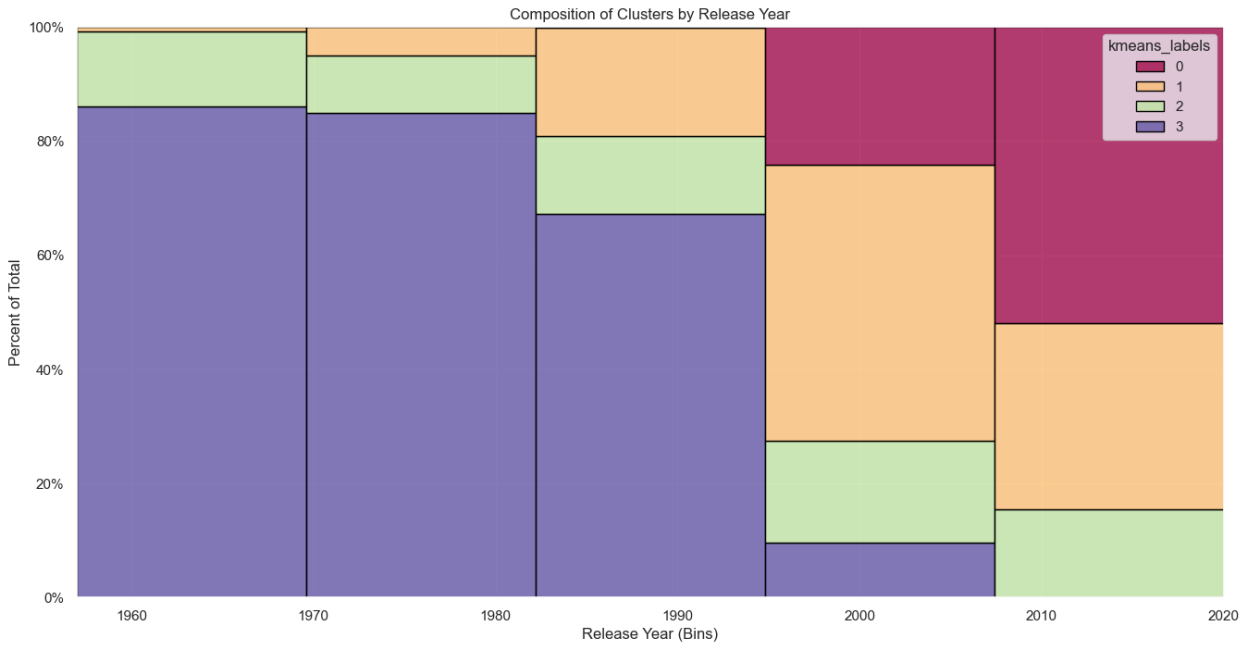
	n_clusters	PCA_Davies_Bouldin	PCA_Calinski_Harabasz	PCA_Silhouette_Score
KMeans Clusters with 4 clusters	4	0.70	34652.0	0.52
Agglomerative Clusters with 4 clusters	4	0.78	31393.0	0.51
KMeans Clusters with 2 clusters	2	0.79	26855.0	0.47
KMeans Clusters with 8 clusters	8	0.76	34550.0	0.46
KMeans Clusters with 14 clusters	14	0.78	34414.0	0.44

After proper scrutinization, *the KMeans clustering algorithm with 4 clusters* shows the second-best distinction and separation amongst clusters (per Calinski-Harabasz score), the most compact clusters with the least overlapping (per Davies-Bouldin index), and the best balance between compactness and cluster separation, demonstrated by samples that are closer to their assigned cluster than to other clusters (per Silhouette score), which deems that models the best performer amongst the models tested.

Cluster Characteristics

Following the application of the best performing clustering algorithm, the resulting clusters were studied to determine the characteristics of each cluster and the context that might be added to the recommendation algorithm was successfully achieved.

Clusters and Year Released

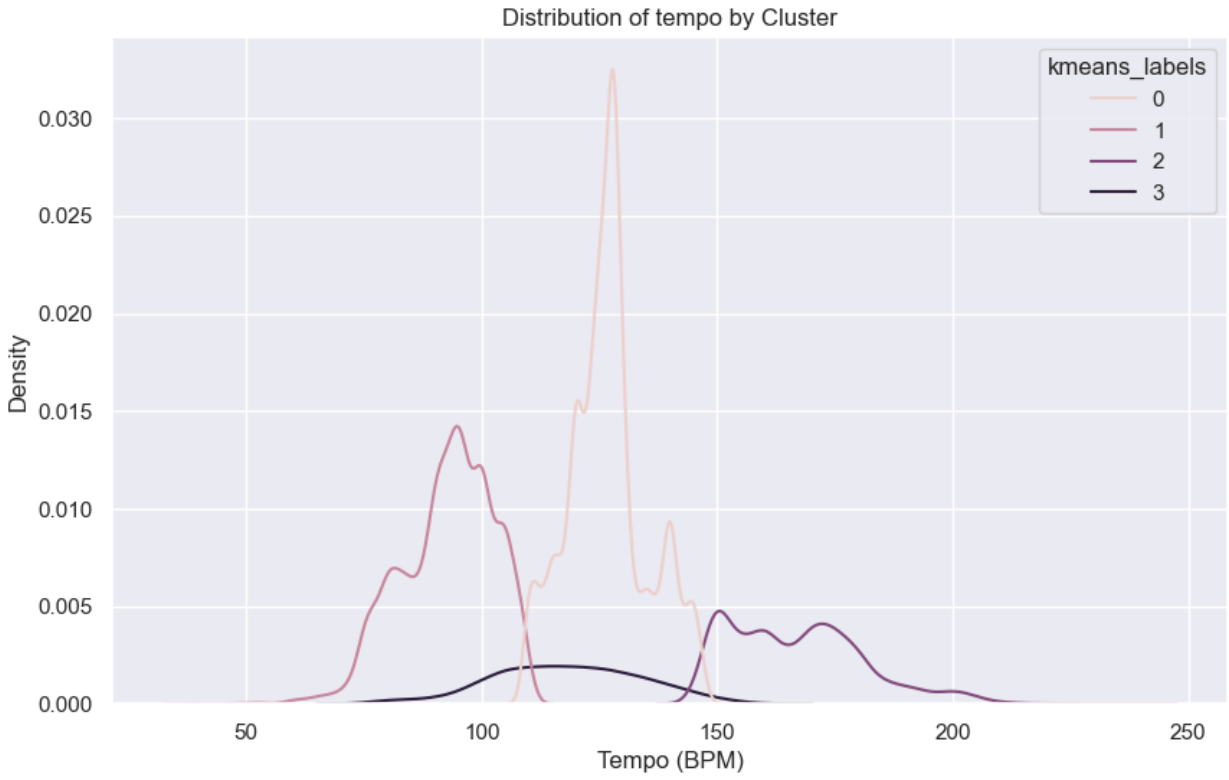


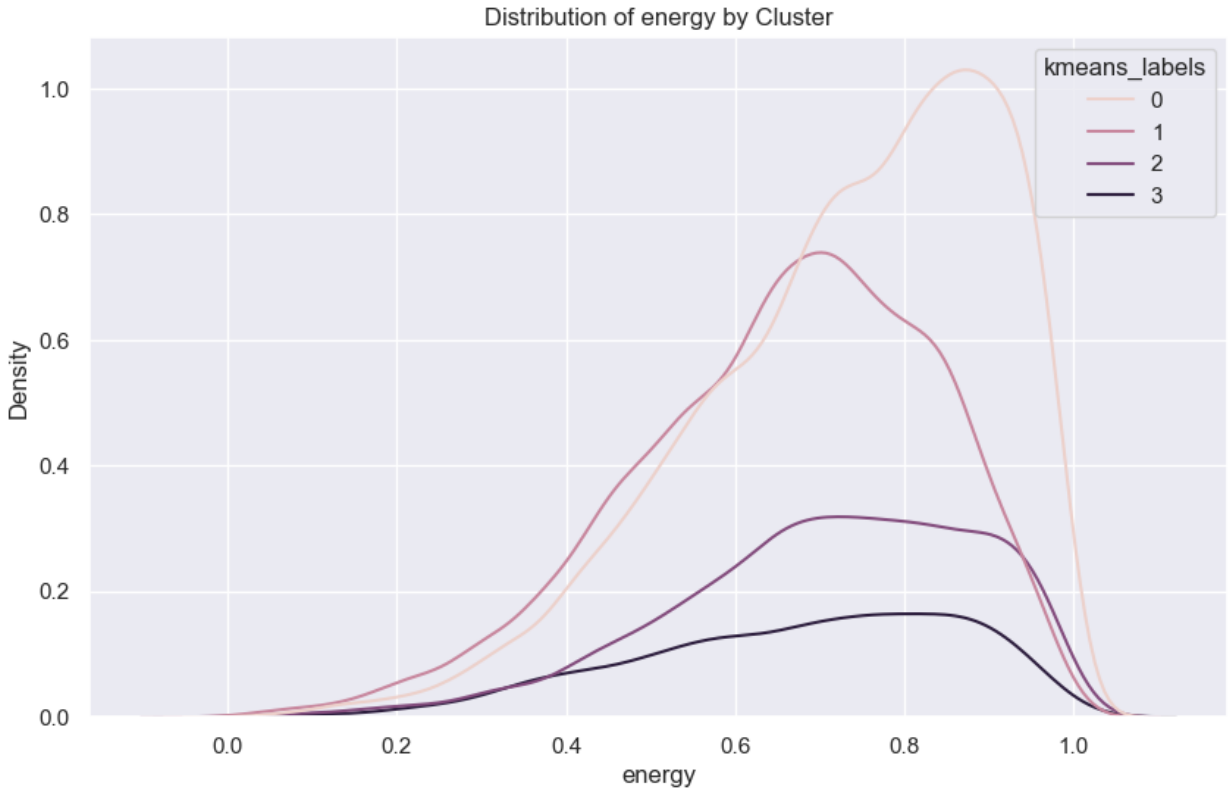
The clustering algorithm appears to have included period context into its clustering logic as follows:

- **Cluster 0:** Composed of newer songs, mostly released between 2010 and 2020.
- **Cluster 1:** Composed of songs mostly released between 2000 and 2010, with some instances between 2010 and 2020 and sporadic instances before the 1990s.
- **Cluster 2:** Fairly evenly distribution across decades.
- **Cluster 3:** Composed of older songs (before the 1990s), with few instances in the 2000s.

Clusters, Tempo and Energy

Another important contextual aspect captured in the clustering process regards the tempo and energy of the tracks and what that relationship may represent. The interpretation of what each cluster might represent can be drawn from observing the distribution of both variables across clusters:



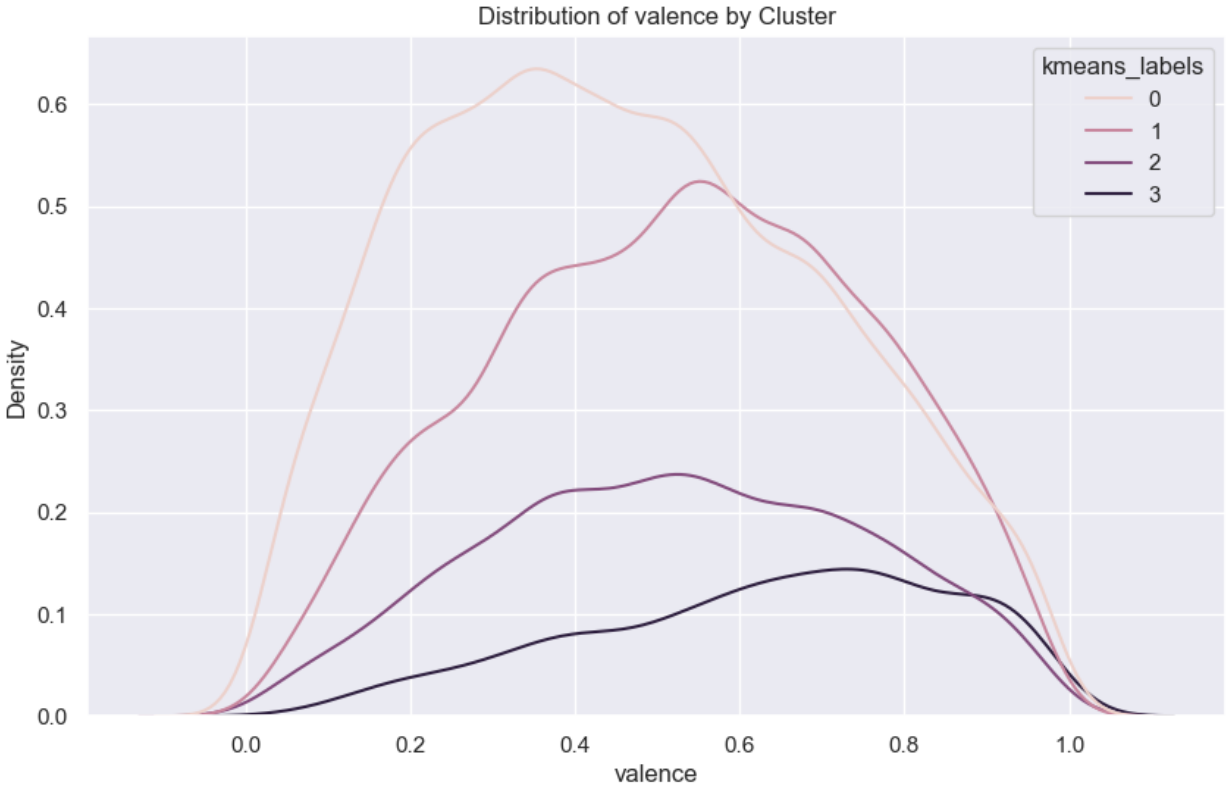


- **Cluster 0**: Composed of songs with moderate tempo (i.e.: not too fast or too slow) with high energy.
- **Cluster 1**: Composed of songs with slow tempo, likely mellow in style, which can be seen by the lowest levels of energy.
- **Cluster 2**: Fastest songs, tending to be moderately to high energy like Folk-pop or Indie Folk music, for example.
- **Cluster 3**: Composed of songs with moderate tempos, although sparser in comparison to cluster 0, which clusters ~125 BPM. More uniformly distributed for energy levels, demonstrating more diversity.

Clusters and Valence

Lastly, an important aspect regarding music is the feeling that it may convey to listeners, which is the definition of the valence variable; music streaming platforms may attempt to extract the feeling conveyed by songs using Convolutional Neural Networks (CNNs), lyrical textual analysis, and other metadata(Chen, 2024) to ensure that the way the listener desires to feel while listening to music matches what the track conveys, further highlighting the importance of valence and mood for

music recommendation.



- **Cluster 0**: Mostly composed of songs with mostly low mood (e.g.: sad, depressed, angry) themes.
- **Cluster 1**: Mostly composed of songs that are mostly neutral in mood and theme.
- **Cluster 2**: Mostly composed of thematically diverse songs, with some tendency for neutrality, though more diverse than cluster 1.
- **Cluster 3**: Mostly composed of songs tending to be thematically upbeat (e.g.: happy, cheerful) in comparison to the remaining clusters, represented by a slight left skew.

Similarity Metrics

The recommendations will be made based on the most similar *n* tracks to the track chosen by the user. Different definitions of similarity will be tested via different similarity metrics for tracks with and without clustering applied to determine what system returns the most appropriate recommendations. As discussed in the *limitations* section of this study, there is no viable alternative to objectively evaluate the recommendations from a user-centric perspective, which may require alternative user-centric evaluation techniques (e.g.: surveys, interviews, and

behavioral data analysis). With that, the recommendations will be evaluated for appropriateness using recognizable songs as an input and ensuring that the recommendations based on that song match in genre, style, feeling evoked, and era.

The similarity of the songs will be evaluated based on characteristics that audibly differentiate one song from the next. For that reason, the **release_year** will not be used to compare similarity metrics since it does not represent a characteristic describing what the song sounds like. Rather it offers context to the algorithm (e.g.: pop from the 1980s vs pop from the 2000s), which has been successfully captured by the clusters. Likewise, the **duration_ms** will not be used to compare the songs either, as that solely describes the length of the track and does not describe any audible characteristic. The characteristics used for similarity metrics are highlighted below:

```
clustering_data =  songs[['danceability', 'energy', 'key', 'loudness', 'mode',
       'speechiness', 'acousticness', 'instrumentalness', 'liveness',
       'valence','tempo']].copy()

```

These features will be tested using **cosine similarity, Mahalanobis distance, and Bray-Curtis dissimilarity.**

Model Evaluation

Test input song: Ozzy Osbourne - Crazy Train

Number of tracks Recommended: 10

Cosine Similarity (documentation)

Cosine similarity measures the cosine of the angle between two non-zero vectors in a multi-dimensional space. It evaluates how aligned two vectors are, irrespective of their magnitude.

The cosine similarity focuses only on the direction of vectors, which would ensure that the songs with similar characteristics are "pointing" to the same direction. In other words, it captures similarity in the "shape" of data points without being affected by their scale(Han & Kamber, 2012). At the same time, magnitude may be important when defining song characteristics, which means that it may not perform optimally in this case. Also, it assumes linear relationships, which may not be accurate for this application.

With Clustering

track_name	track_artist	track_album_name
Spicy (with Diplo & Charli XCX)	Herve Pagez	Spicy (with Diplo & Charli XCX)
光線	Ghost like girlfriend	光線
My One (feat. Tory Lanez, Kranium & Dappy) (To...	Wiley	My One (Todd Edwards Remix)
Bésame	David Bisbal	Bésame
Paradise	Matthew Parker	Daydreamer
Grateful (Neon Feather Remix)	Cade Thompson	Grateful (Neon Feather Remix)
Nothin' to Compare	Who	Nothin' to Compare
What Am I	Why Don't We	What Am I
It's Time	Imagine Dragons	Night Visions
Poppin' Champagne	All Time Low	So Wrong, It's Right (Deluxe Version)

Without Clustering

track_name	track_artist	track_album_name
Big in Japan - Best of-Version	Alphaville	First Harvest 1984-1992
Y Ahora	Nengo Flow	Y Ahora
Uprising	Muse	The Resistance
Cool Cat - Remastered 2011	Queen	Hot Space (2011 Remaster)
Hombre Al Agua - Remasterizado 2007	Soda Stereo	El Ultimo Concierto A
Barracuda	Heart	Little Queen
Lost In Love	Air Supply	Lost in Love
Walkin' On The Sun	Smash Mouth	Fush Yu Mang (20th Anniversary Edition)
Summer Of '69	Bryan Adams	Anthology (set - UK)
4 Life (feat. Graham Candy)	Robin Schulz	Sugar

Mahalanobis Distance (documentation**)**

The Mahalanobis distance measures the distance between a point and a distribution, considering correlations between features. Because the distance from the point is in relation to distributions, correlations are accounted for in the covariance matrix and are independent of scales. These two characteristics are especially good for a dataset in which so few PCA components (2) can explain so much of the variance, as this may indicate close correlation between variables.

The Mahalanobis distance requires the inverse of the covariance matrix, which may not be inversible because, among other factors, the dataset has linearly dependent features or if the number of features exceeds the number of observations and is the case in this application. When that is the case, the pseudoinverse (Moore-Penrose inverse) will be used instead of the inverse, as it is often used in linear algebra problems where the standard inverse cannot be applied(Kumar, 2022). This approach may introduce small inaccuracies in the process when compared to the true inverse, if the true inverse existent.

With Clustering

track_name	track_artist	track_album_name
Hair Of The Dog	Guns N' Roses	The Spaghetti Incident?
All Day and All of the Night	The Kinks	Kinks (Deluxe Edition)
Tonight's the Night (Gonna Be Alright)	Rod Stewart	A Night on the Town
Keep Yourself Alive - 2011 Mix	Queen	Queen (2011 Remaster)
Need Your Loving Tonight - Remastered 2011	Queen	The Game (2011 Remaster)
Dance The Night Away	Van Halen	Van Halen II
Panama - 2015 Remaster	Van Halen	1984 (Remastered)
Stay Free - Remastered	The Clash	Give 'Em Enough Rope (Remastered)
Bad Boys Running Wild	Scorpions	Love At First Sting (50th Anniversary Deluxe E...
Seek & Destroy	Metallica	Kill 'Em All

Without Clustering

track_name	track_artist	track_album_name
Pleaser	Wallows	Pleaser
Echo	Bad Meets Evil	Hell: The Sequel (Deluxe)
All Day and All of the Night	The Kinks	Kinks (Deluxe Edition)
Killer Queen - Remastered 2011	Queen	Sheer Heart Attack (2011 Remaster)
Killer Queen - 2011 Mix	Queen	Sheer Heart Attack (2011 Remaster)
Get Down, Make Love - Remastered 2011	Queen	News Of The World (2011 Remaster)
Need Your Loving Tonight - Remastered 2011	Queen	The Game (2011 Remaster)
If You're Gone	Matchbox Twenty	Mad Season
Bad Boys Running Wild	Scorpions	Love At First Sting (50th Anniversary Deluxe E...
Seek & Destroy	Metallica	Kill 'Em All

Bray-Curtis Dissimilarity Score (documentation)

The Bray-Curtis dissimilarity is a metric used to quantify the compositional difference between two samples based on their feature abundances, ranging from 0 (identical) to 1 (completely dissimilar). This measure would normalize by the sum of features, making it robust to differences in total counts or, in this case, magnitudes(Bakker, 2024). Because this metric deals with abundance and composition, negative values are not well tolerated by the Bray-Curtis dissimilarity metric; for that reason, the MinMaxScaler will be applied to the loudness feature to limit it from 0 to 1, thus eliminating negative values. Unlike the Mahalanobis metric it cannot account for relationships between features.

With Clustering

track_name	track_artist	track_album_name
Right Next Door To Hell	Guns N' Roses	Use Your Illusion I
最後のナイト・フライト	オメガトライブ	River's Island
The Temple Of The King	Rainbow	Ritchie Blackmore's Rainbow
Good Golly Miss Molly	Creedence Clearwater Revival	Bayou Country (40th Anniversary Edition)
Heaven	Ebo Taylor	Life Stories
Stairway to Heaven - Remaster	Led Zeppelin	Led Zeppelin IV (Deluxe Edition)
Holy Diver	Dio	Holy Diver
Disorder - 2007 Remaster	Joy Division	Unknown Pleasures
Billion Dollar Babies	Alice Cooper	Alice Cooper's Greatest Hits
Hard Times	Baby Huey & The Baby Sitters	The Baby Huey Story: The Living Legend

Without Clustering

track_name	track_artist	track_album_name
You Send Me	Nicolette Larson	Nicolette
The Temple Of The King	Rainbow	Ritchie Blackmore's Rainbow
Good Golly Miss Molly	Creedence Clearwater Revival	Bayou Country (40th Anniversary Edition)
Heaven	Ebo Taylor	Life Stories
Stairway to Heaven - Remaster	Led Zeppelin	Led Zeppelin IV (Deluxe Edition)
Holy Diver	Dio	Holy Diver
Disorder - 2007 Remaster	Joy Division	Unknown Pleasures
Vuelvo	Cami	Monstruo (Pt. 1)
Hard Times	Baby Huey & The Baby Sitters	The Baby Huey Story: The Living Legend
Monsters in Your Bedroom	Tertia May	Monsters in Your Bedroom

Production Model

The recommendations output by the models were compared and evaluated for reasonableness and appropriateness as shown in the *Model Evaluation* section above for diverse tracks spanning multiple genres, decades, and musical styles to ensure good, generalized performance defines as aligned genre, style, feeling evoked, and era similarity in comparison to the input track. Based on those parameters, the recommendations for the tracks below were evaluated to establish the best similarity metric and clustering/no clustering combination:

- Ozzy Osbourne - Crazy Train
- Adele – Someone Like You
- Luke Bryan – Country Girl (Shake It For Me)
- Taylor Swift – Shake it Off
- Girls Just Want to Have Fun – Cindy Lauper
- The Black Keys - Tighten Up

The empirical study conducted indicated that the clustering algorithm added important context and limited the comparisons to calculate the similarity metrics to more relevant tracks, yielding better recommendations and less comparisons to be carried out. For those reasons, the clustering algorithm is deemed essential for the production model.

In addition to that, the Mahalanobis distance was the metric that yielded the most appropriate recommendations, often capturing nuances that the remaining metrics did not appear to capture, with the Bray-Curtis following behind and the cosine similarity performing the worst. The ability of the Mahalanobis distance to account for correlations is believed to have been the differential that crowned this metric as the best performer, given that the low component-high explained variation found in the PCA might indicate multicollinearity and redundancy to be present in the song features. Lastly, the fact that the Mahalanobis distance calculates the distance between one point

and a distribution likely helped with the generalization, given that the song characteristics must be interpreted holistically to determine the genre, era, musical style, and other song characteristics. Comparing points to distributions likely aided in generalizing distributions that represent these collective characteristics better, thus yielding better recommendations.

With that, the model that uses the Mahalanobis distance as a basis for similarity with the PCA and KMeans clustering algorithm applied will move forward for production and deployment. This is expected to provide straightforward, user-centric recommendations utilizing a content-based recommendation algorithm as an alternative to current approaches taken by music streaming platforms.

In addition to this model, a secondary, much simpler model will also be pushed into production. A heuristic approach was used to create a secondary model that uses energy and valence to determine the moods and feelings evoked by songs. This is a much more generalized algorithm expected to north undecided listeners that may not have a define track in mind to be used with the primary algorithm by giving them suggestions that could later be enjoyed or used as the input for the primary algorithm. This is also expected to be a simple path for users that might not be as rigorous with their current musical needs to receive recommendations.

Model Deployment

The deployment of the proposed recommendation system involves hosting it as a web application accessible to users for generating music recommendations based on their input. This section outlines the deployment steps, including the technology stack and frameworks utilized.

Deployment Platform and Framework

The recommendation system is deployed using Streamlit, an open-source Python framework for building interactive and user-friendly web applications. Streamlit was chosen due to its simplicity in integrating machine learning models with a clean user interface.

Authentication and Integration with Spotify API

To access Spotify's data and manage user playlists, the application uses the Spotipy library, a lightweight Python client for the Spotify Web API. The OAuth2 authentication process enables the

app to interact securely with user-specific resources, including retrieving user IDs and managing playlists.

The components that are essential for deployment include *Spotify Developer Account*, which are the credentials (client ID and secret) are obtained by registering the app in the Spotify Developer Dashboard, and authentication Handling with OAuth2, which is managed using SpotifyOAuth from Spotipy. This ensures users authenticate securely, and tokens are refreshed as needed to maintain session integrity.

Web Application Features

The deployed application offers two main functionalities:

1. **Music Recommendations**: Users can input preferences, such as selecting specific tracks or defining a mood, to receive personalized music recommendations. These recommendations are generated using pre-trained clustering models and similarity metrics, which are integrated seamlessly into the web interface.
2. **Playlist Management**: Users can save recommended tracks directly to new or existing playlists in their Spotify library. The application ensures playlist management is efficient and aligns with Spotify's user data permissions.

Streamlit's interactive widgets allowed for real-time input processing, while the Spotify API ensured a seamless connection to user accounts.

Implementation Steps

1. **Setting Up Dependencies**:
    - The Python environment includes the spotipy, numpy, pandas, and Streamlit libraries.
    - The Streamlit app integrates the machine learning models and Spotify API functionalities seamlessly.
2. **Authentication Flow**:
    - The OAuth2 flow is triggered with SpotifyOAuth, generating a unique token for each session.
    - User credentials are validated, and session states are managed using st.session_stateto persist user-specific data.
3. **Recommendation Engine**:

- The clustering and similarity metrics implemented during development are integrated into the web application.
- Real-time user input dynamically updates recommendations displayed in an interactive interface.

4. **Deployment**:
    - The application is hosted on **Streamlit Cloud** for public accessibility. The deployment ensures scalability and ease of updates.

The deployed application demonstrates a streamlined, user-focused approach to music recommendations. By leveraging Spotify's API and Streamlit's framework, the solution achieves accessibility, functionality, and scalability. Future enhancements could include expanding data sources for recommendations and integrating more advanced model updates seamlessly into the deployment pipeline.

Summary and Conclusions

The core idea of this study was to develop an alternative to the approach currently presented by music streaming platforms, especially Spotify, as a recommendation system. With the massification of machine learning and artificial intelligence applications, these recommendations algorithms have increasingly become more complex to optimize the user experience within the application, and as consequence, retain users active in the platform. Despite the increasing complexity and search for the best experience, users still report unsatisfactory experiences with the recommendation system. Given the dissonance between increased complexity and unsatisfactory results, this project decided to go in the opposite direction by offering a simpler, less computationally expensive algorithm that focused on the user and the content that the user desired to consume at that time. By lessening the assumptions about what was thought to be that the user wanted and taking a user-centric approach by asking the user for inputs instead, we expect to output recommendations that are more closely aligned to the user and how they feel. The outcome of that goal was to develop and deploy the SimplyFy Web Application, an interactive tool that allows users to share and experience the fundaments of this study in practice.

Final Recommendations and Future Enhancements

Although the SimplyFy Web Application was successfully deploy and meets the initial goal of the initial scope of this study, which was delivering a simple, efficient, and user-centric content-based music recommendation algorithm, we strongly believe in an iterative approach for this tool. As highlighted in the study, the tool currently relies on an "offline" database which limits the pool of recommendations for users. Connecting directly to Spotify's API would be a likely step to follow in future iterations of the tool. Additionally, recommendation algorithms are only as good as users deem their performance to be; getting more user feedback to develop the layout, UX, and tuning the overall recommendation performance of the algorithm is essential for a tool like SimplyFy to be successful in delivering an optimal experience to users. With that, this likely will be a tool that will be built upon to include more functionality and improve performance.

References

Bakker, J. (2024). *Applied Multivariate Statistics in R*. University of Washington. https://uw.pressbooks.pub/appliedmultivariatestatistics/

Chen, Y. (2024). Music recommendation systems in music information retrieval: Leveraging machine learning and data mining techniques. *Applied and Computational Engineering*, *87*(1), 197–202. https://doi.org/10.54254/2755-2721/87/20241564

De Assunção, W. G., & Zaina, L. A. M. (2022). Evaluating user experience in music discovery on deezer and spotify. *Proceedings of the 21st Brazilian Symposium on Human Factors in Computing Systems*, 1–11. https://doi.org/10.1145/3554364.3560901

Han, J., & Kamber, M. (2012). *Data mining: Concepts and techniques* (3rd ed). Elsevier.

Hodgson, T. (2021). Spotify and the democratisation of music. *Popular Music*, *40*(1), 1–17. https://doi.org/10.1017/S0261143021000064

Jr, R. M. S., Wei, D., Benson, N., & Javed, F. (2021). *Alternative Methods for Deriving Emotion Metrics in the Spotify® Recommendation Algorithm*. *5*(3).

Kumar, M. (2022, June 21). Moore–Penrose Inverse Simplified! *Medium*. https://medium.com/accredian/moore-penrose-inverse-simplified-3e0a6cd2965e

Ng, T. (2024, August 16). How to break free of Spotify's algorithm. *MIT Technology Review*. https://www.technologyreview.com/2024/08/16/1096276/spotify-algorithms-music-discovery-ux/

Schedl, M., Zamani, H., Chen, C.-W., Deldjoo, Y., & Elahi, M. (2018). Current challenges and visions in music recommender systems research. *International Journal of Multimedia Information Retrieval*, *7*(2), 95–116. https://doi.org/10.1007/s13735-018-0154-2

Spotify. (n.d.). *Company Info*. https://newsroom.spotify.com/company-info/

Yahoo Finance. (n.d.). *Spotify Technology S.A. (SPOT)*. Yahoo Finance. Retrieved November 16, 2024, from https://finance.yahoo.com/quote/SPOT/