SC1015 MINI PROJECT

HOW TO MAKE SUCCESSFUL SONGS ON SPOTIFY

Ooi Shania U2322533D
Tan Jing Wen, Kellie U2320009J
Sabareesh Kannan U2322638L

PROBLEM DEFINITION

To utilise machine learning algorithms to predict the success of songs through the number of streams on Spotify. This is done by analysing a wide range of input variables, including musical and non-musical features like instrumentalness, valence and release date.

MOTIVATIONS

- Optimise marketing strategies for music labels
- Provide targeted feedback to artists
- Offer valuable insights that enable stakeholders to navigate the complex landscape of the music industry more effectively

EXPLORATORY DATA ANALYSIS

DATASET: TOP SPOTIFY SONGS

Top Spotify Songs
by Kumar Arnav

‘Streams’ Column

Maximum	3,703,895,000
Minimum	2762
Mean	514,137,400
Median	290,530,900
Standard Deviation	566,856,900

https://www.kaggle.com/datasets/arnavvvvv/spotify-music?resource=download

DATA CLEANING

- Remove non-numeric 'streams' values

```
data_raw['streams'] = pd.to_numeric(data_raw['streams'], errors="coerce", downcast='integer')
data_raw.dropna(subset=['streams'], inplace=True)
data_raw
```

- Remove duplicates of 'track_name'
- Sort in ascending order of ‘streams’

```
data = data_raw.drop_duplicates('track_name').sort_values('streams')
data
```

DATA CLEANING

- Remove irrelevant columns

```
irr_cols = ['artist_count', 'released_day', 'in_apple_playlists', 'in_apple_charts',
            'in_deezer_charts', 'in_deezer_playlists', 'in_shazam_charts', 'key']
data.drop(columns=irr_cols, inplace=True)
data.head()
```

- Check for null values to remove

```
for i in data:
    nulls = data[i].isnull().sum()
    print(f"Nulls for {i}: {nulls}")
```

DATA CLEANING

Description of Relevant Features

Features:

track_name: *name of the song*

artist(s)_name: *name of the artist*

released_year: *year the song is released* --> numerical

released_month: *month the song is released* --> categorical

```
MONTHS OF A YEAR:
1    JAN
2    FEB
3    MAR
...
```

in_spotify_playlists: *number of spotify playlists the songs are in* --> numerical

in_spotify_charts: *number of spotify charts the songs are in* --> numerical

streams: *number of times the songs are played* --> numerical

bpm: "*beats per minute*", *tempo of the song* --> numerical

mode: *type of musical scale* --> categorical

```
MAJOR
MINOR
```

MUSICAL FEATURES ANALYSIS

- Selected relevant musical features for analysis

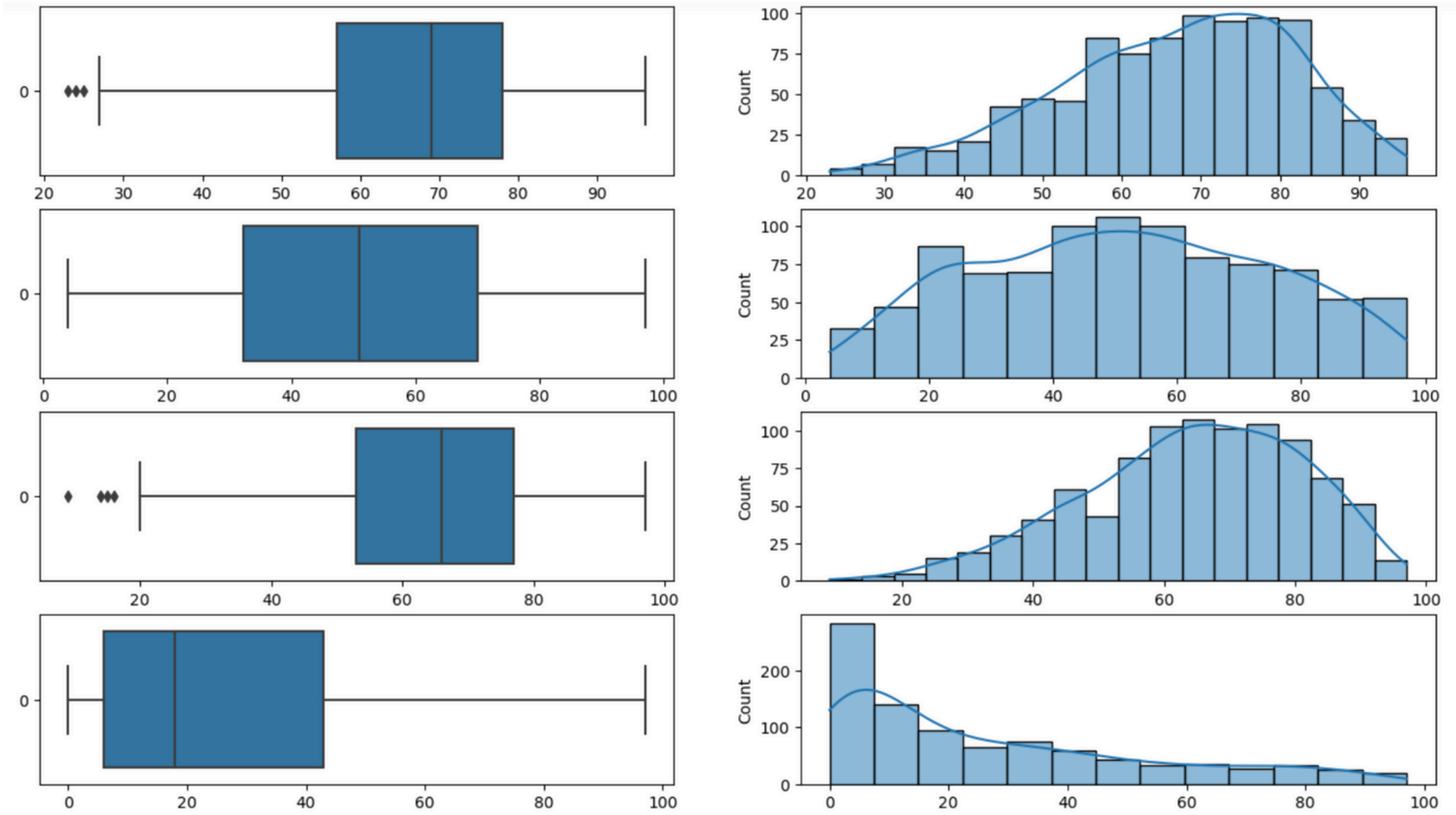
In [36]:
```
musical_features = ['danceability_%', 'valence_%', 'energy_%', 'acousticness_%',
                    'instrumentalness_%', 'liveness_%', 'speechiness_%']

musical_data = clean_data[musical_features].copy()
musical_data.head()
```

Out[36]:

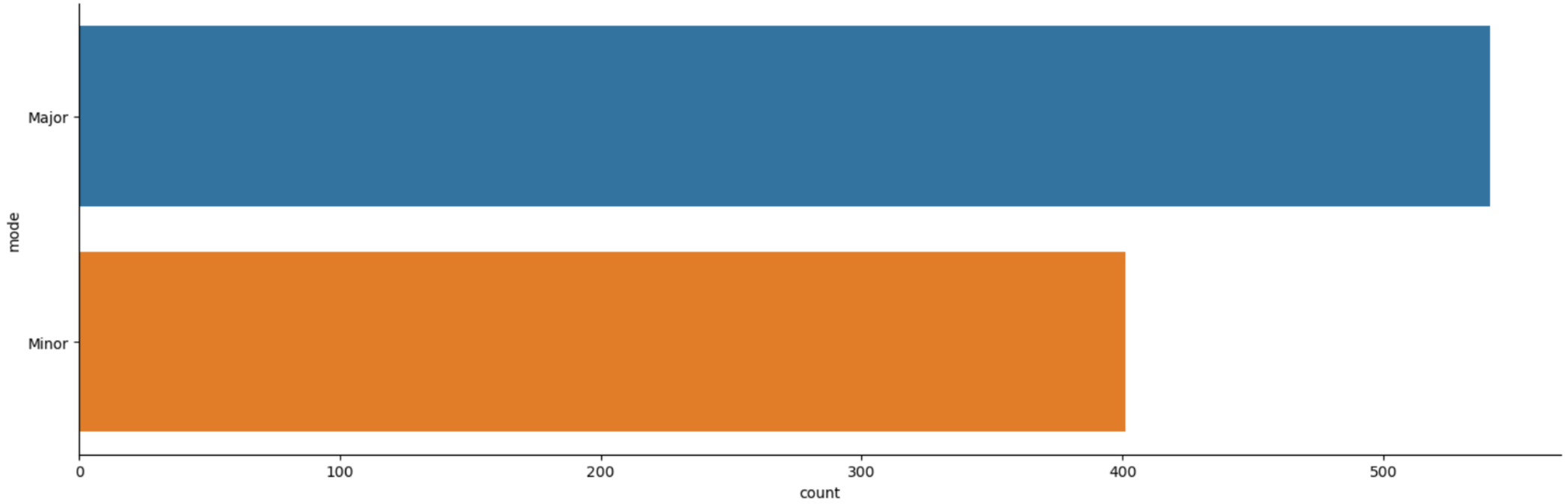
	danceability_%	valence_%	energy_%	acousticness_%	instrumentalness_%	liveness_%	speechiness_%
123	49	78	64	19	0	11	4
393	82	62	74	10	0	33	7

MUSICAL FEATURES ANALYSIS

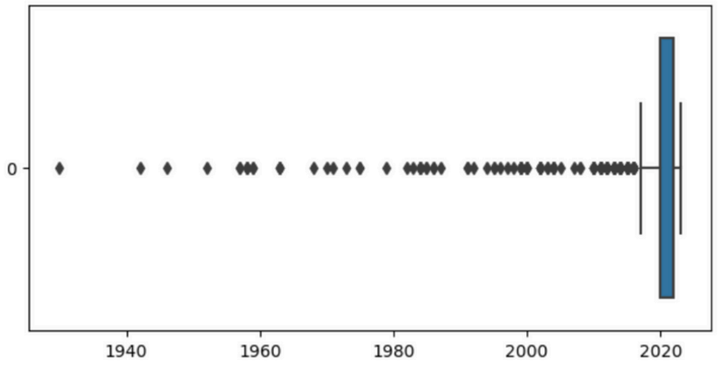


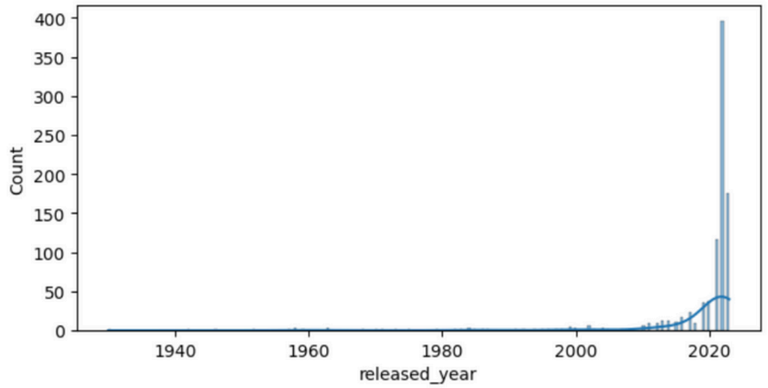
MUSICAL FEATURES ANALYSIS

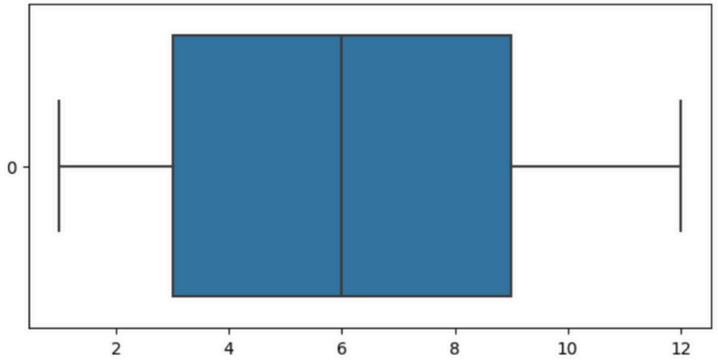
- Catplot of ‘mode’ shows a preference for songs with mode ‘Major’ , indicating that most popular songs tend to be in a major key.

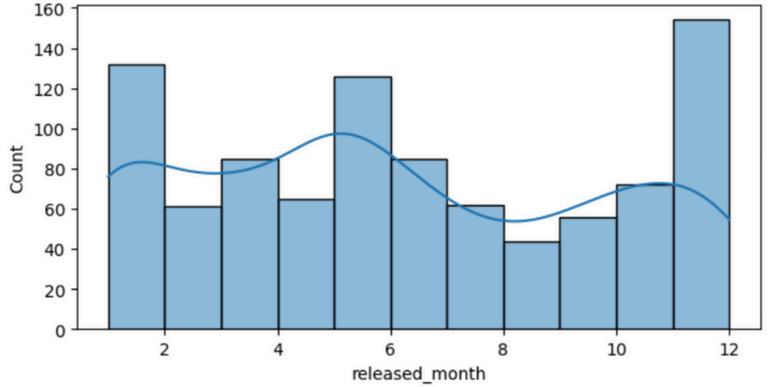


NON-MUSICAL FEATURES ANALYSIS



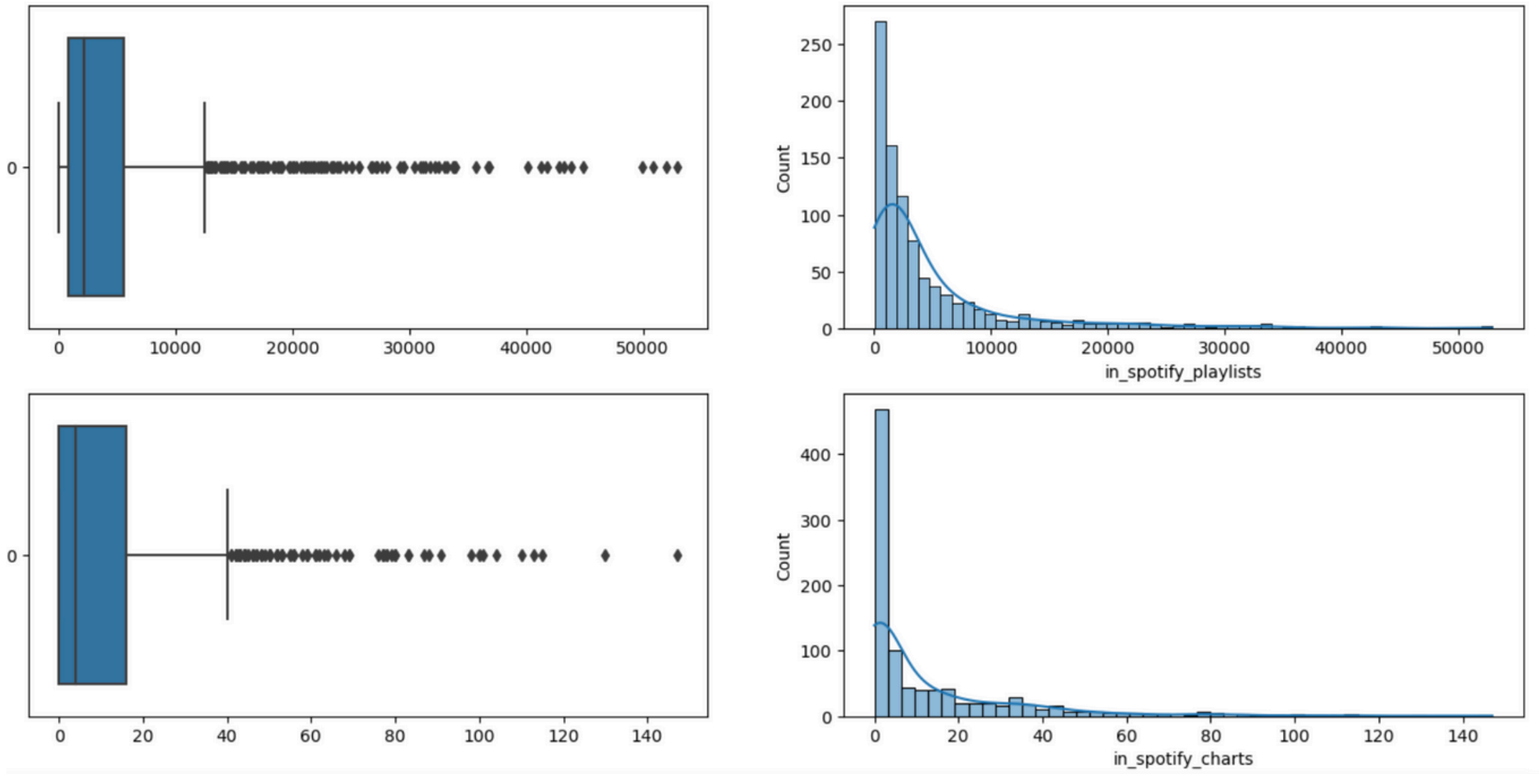






- **released_year**: a range of years, possibly indicating a dataset covering multiple years
- **released_month:** a uniform distribution, possibly indicating equal representation of all months

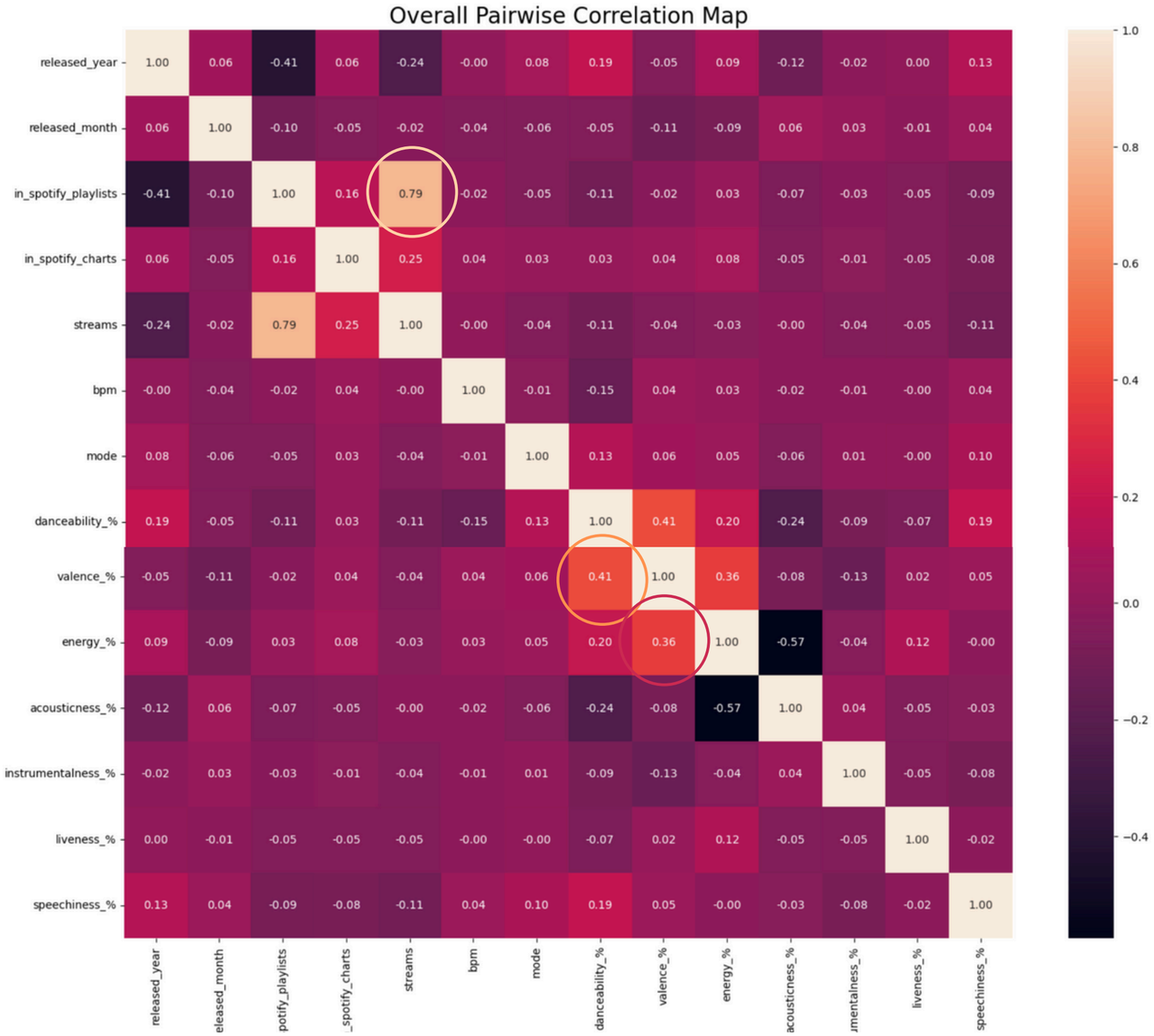
NON-MUSICAL FEATURES ANALYSIS



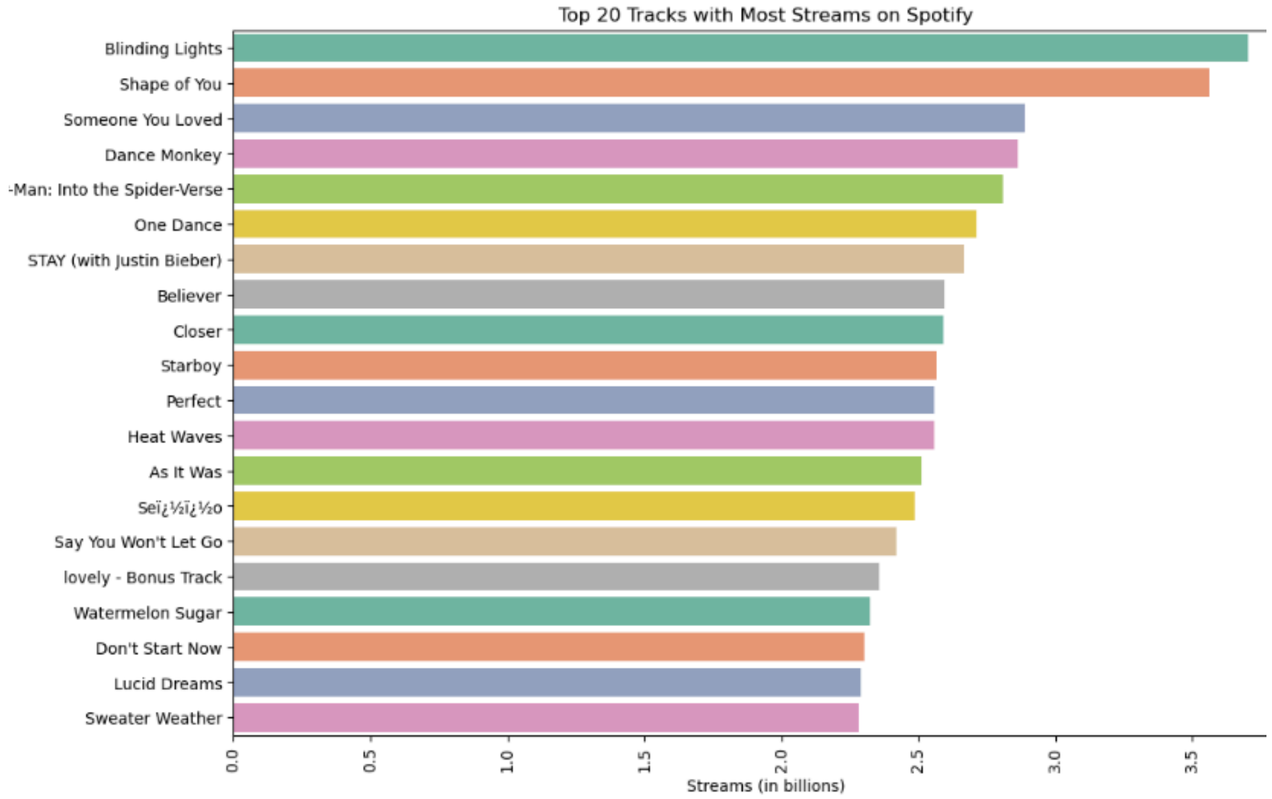
- **in_spotify_playlists**: whether a song is included in a spotify playlist, indicating most songs are only in a few playlists
- **in_spotify_charts:** whether a song is included in a spotify chart, indicating most songs are not in charts

ALL FEATURES

- Reveals strong correlations between certain features, indicating potential relationships and patterns in the data.
- “streams” and “in_spotify_ playlists” have a strong correlation of **0.79**.
- “valence_%” has a moderate correlation between “danceability_%” and “energy_%”, with values of **0.41** and **0.39** respectively.



TOP 20 (MOST STREAMS)



Top 20 Tracks by Top 20 Artists

```
pd.merge(top20_tracks, top20_artists, on="artist(s)_name", how="i
```

	track_name	artist(s)_name	track streams	artist streams
0	Blinding Lights	The Weeknd	3.703895e+09	1.375285e+10
1	Shape of You	Ed Sheeran	3.562544e+09	1.390895e+10
2	Perfect	Ed Sheeran	2.559529e+09	1.390895e+10
3	Someone You Loved	Lewis Capaldi	2.887242e+09	4.734698e+09
4	Believer	Imagine Dragons	2.594040e+09	5.272485e+09
5	As It Was	Harry Styles	2.513188e+09	1.131781e+10
6	Watermelon Sugar	Harry Styles	2.322580e+09	1.131781e+10
7	Don't Start Now	Dua Lipa	2.303034e+09	3.227639e+09
8	Sweater Weather	The Neighbourhood	2.282771e+09	4.010010e+09

8 of the Top 20 tracks are by established artists, whereas the rest of the tracks are by newer and upcoming artists.

TOP 20 (ANALYSIS)

Top 20 Artists with Most Streams on Spotify

In [18]:
```
top20_artists = data.groupby(["artist(s)_name"]).streams.sum().reset_index().sort_values(by='streams', ascending=Fal
top20_artists
```

Top 20 Tracks in Users' Spotify Playlists

In [19]:
```
top20_in_spotify_playlist = data[['track_name', 'in_spotify_playlists']].sort_values(by='in_spotify_playlists', asce
top20_in_spotify_playlist
```

Top 20 Tracks in Spotify Charts

In [20]:
```
top20_in_spotify_charts = data[['track_name', 'in_spotify_charts']].sort_values(by='in_spotify_charts', ascending=Fa
top20_in_spotify_charts
```

Top 20 Tracks by Top 20 Artists

In [21]:
```
pd.merge(top20_tracks, top20_artists, on="artist(s)_name", how="inner").rename(columns={"streams_x": "track streams"
```

Top 20 Tracks in Top 20 Tracks in Spotify Playlists

In [22]:
```
pd.merge(top20_tracks, top20_in_spotify_playlist, on="track_name", how="inner")
```

- Top 20 Artists: A mix of established and newer artists, supporting the notion that both types of artists can achieve success on Spotify.
- Playlist and Chart Features: The top 20 tracks in Spotify playlists and charts are mostly different from the top 20 tracks by streams, indicating that playlist and chart features may be influenced by different factors than overall popularity.

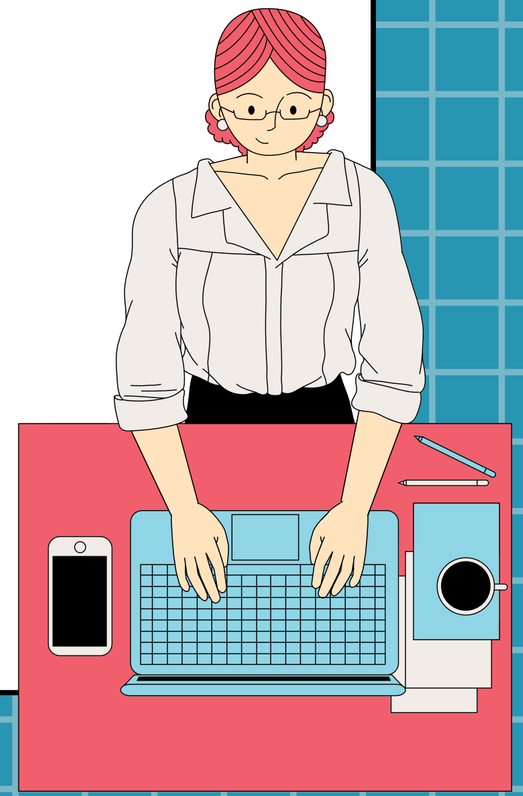
MACHINE LEARNING

Clustering with K-means Algorithm

K-MEANS CLUSTERING

- Unsupervised Machine Learing Algorithm
- Provides a data-driven approach to predicting song success

1. Identify key features that contribute to a song's success
2. Classify new songs into these clusters based on their musical features
3. Predict their likelihood of success



K-MEANS CLUSTERING

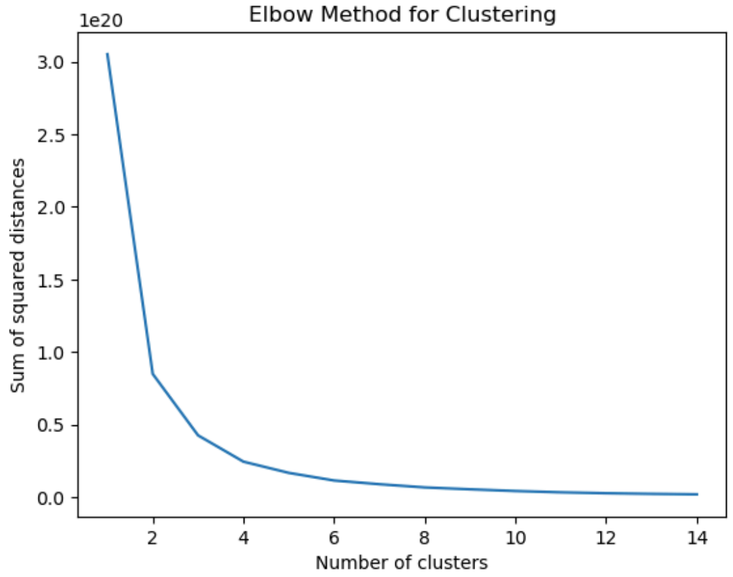
1. Initialise KMeans

```
sse = [] # Empty list to store the sum of squared distances for each number of clusters

# Fit KMeans model to the data with a range of different numbers of clusters
for k in range(1, 15):
    kmeans = KMeans(n_clusters=k, n_init=10)
    kmeans.fit(numeric_data)
    sse.append(kmeans.inertia_)
    # Add the sum of squared distances for the current number of clusters to the list
```

K-MEANS CLUSTERING

1. Initialise KMeans
2. Using Elbow Method, determine optimal number of clusters




```
# Plot the sum squared distances for each number of clusters for elbow method

plt.plot(range(1, 15), sse)
plt.title('Elbow Method for Clustering')
plt.xlabel('Number of clusters')
plt.ylabel('Sum of squared distances')
plt.show()
```

K-MEANS CLUSTERING

1. Initialise KMeans
2. Using Elbow Method, determine optimal number of clusters
3. Generate Clusters

```
clusters = kmeans.predict(numeric_data) # Generate cluster assignments for each data point
```

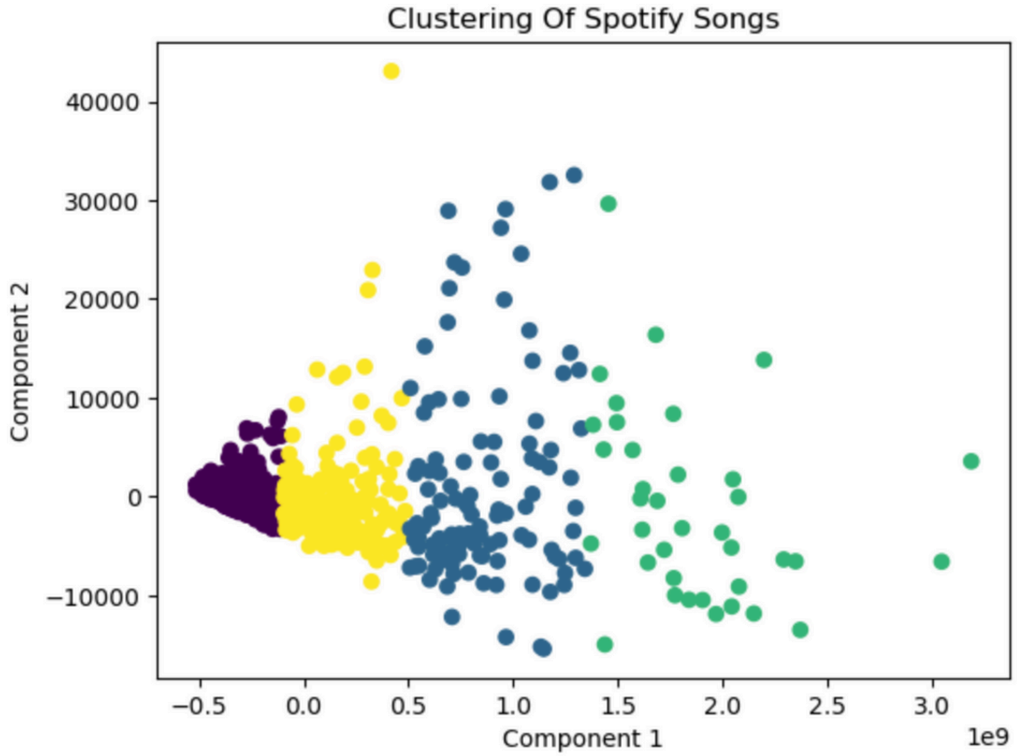
K-MEANS CLUSTERING

1. Initialise KMeans
2. Using Elbow Method, determine optimal number of clusters
3. Generate Clusters
4. Evaluate Quality of Clusters

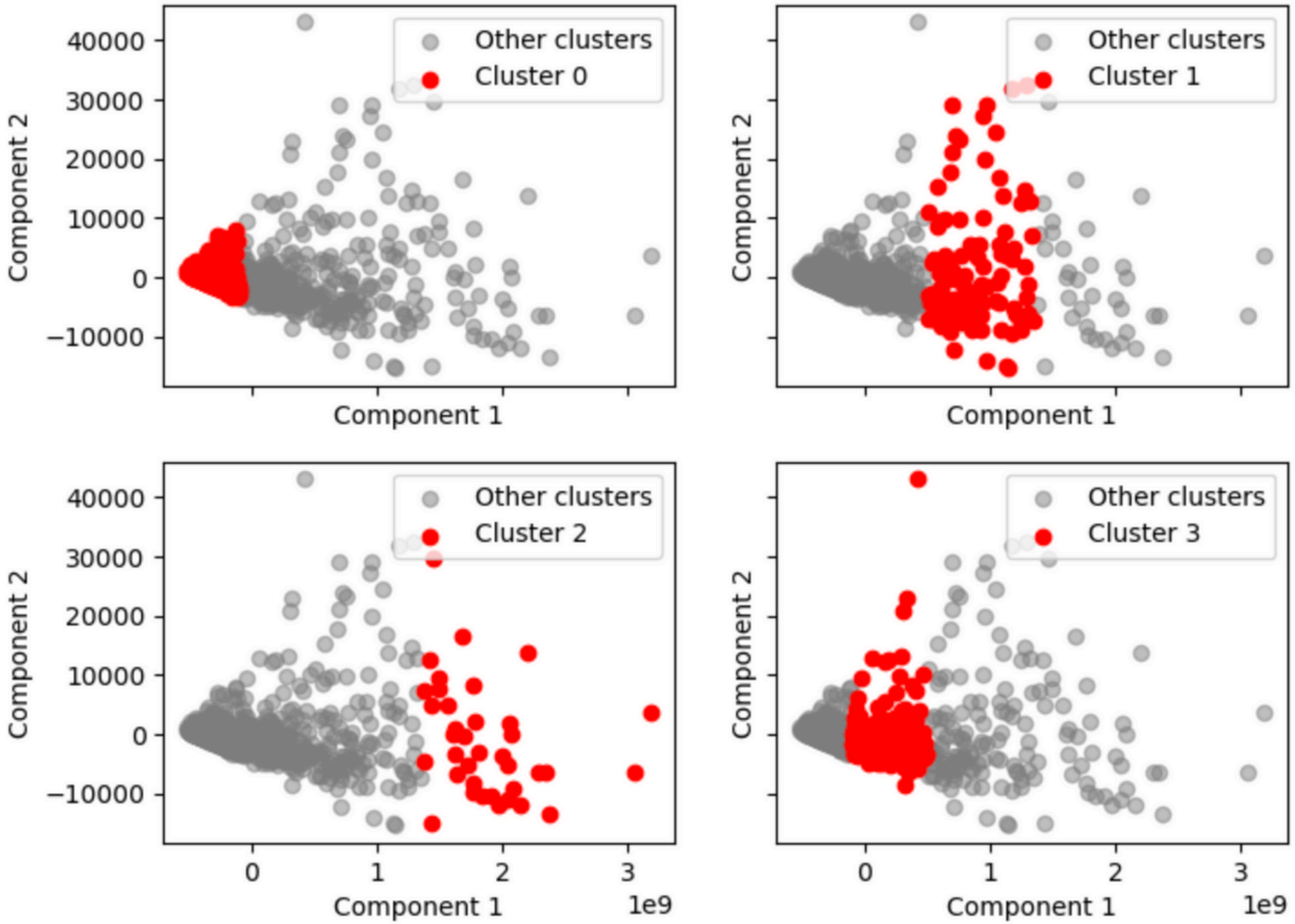
```
# Calculate the silhouette score for the generated clusters
silhouette_score(numeric_data, clusters)
```

```
0.6403856789309437
```

VISUALIZE THE GENERATED CLUSTERS



VISUALIZE THE GENERATED CLUSTERS



CLUSTER CHARACTERISTICS

- Helps us gain insights into the characteristics of each cluster and how they relate to song success

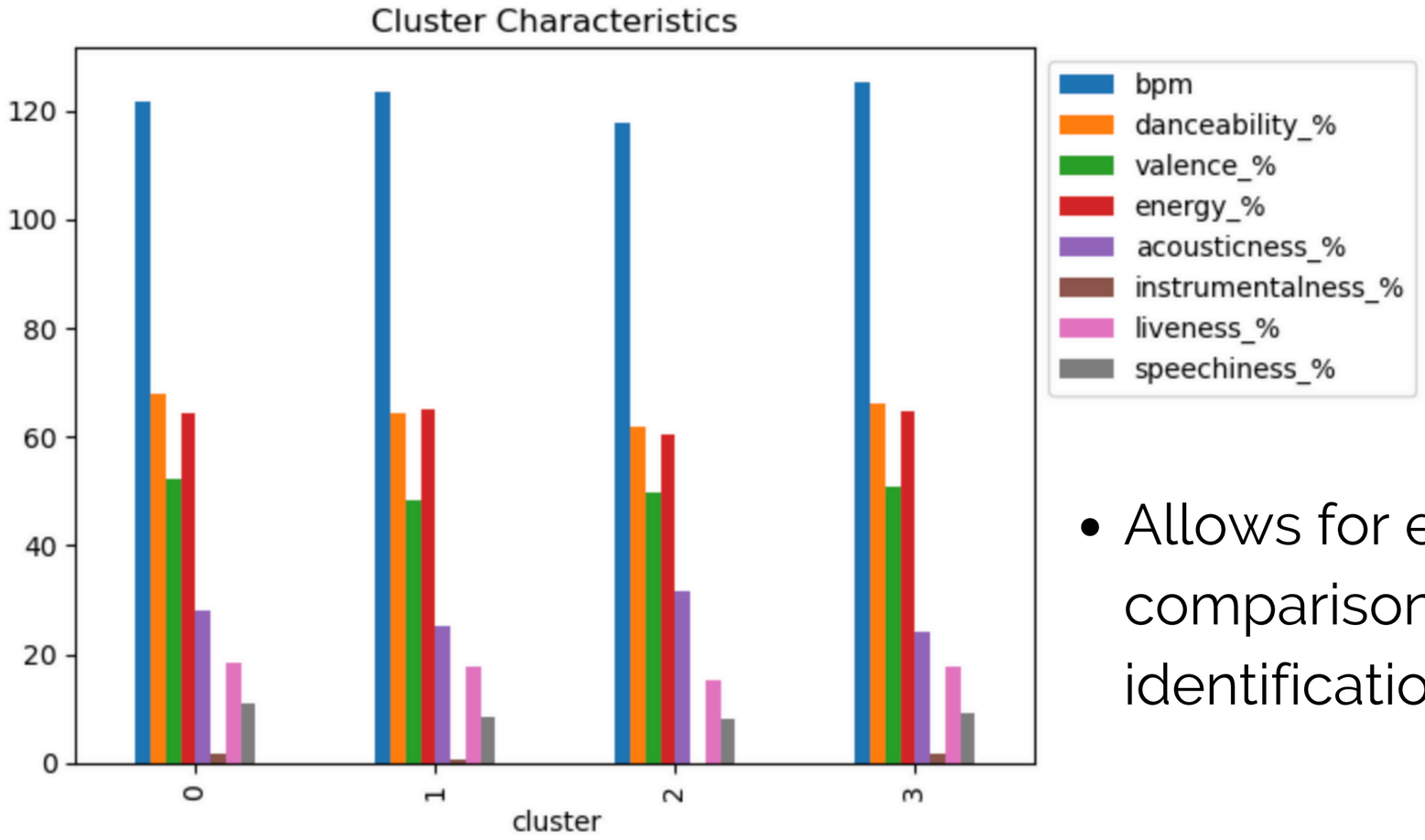
```
key_features = data[['bpm', 'danceability_%', 'valence_%', 'energy_%', 'acousticness_%',
                     'instrumentalness_%', 'liveness_%', 'speechiness_%']]
clustered_df = key_features.copy()
clustered_df['cluster'] = clusters

cluster_means = clustered_df.groupby('cluster').mean()
print(cluster_means)
```

```
                bpm  danceability_%  valence_%   energy_%  acousticness_%  \
cluster
0        121.490662       68.057725  52.327674  64.337861       28.108659
1        123.333333       64.228070  48.333333  65.000000       25.298246
2        117.540541       62.027027  49.891892  60.513514       31.621622
3        125.237624       66.227723  50.722772  64.693069       24.009901

         instrumentalness_%  liveness_%  speechiness_%
cluster
0                  1.820034   18.601019      10.964346
1                  0.675439   17.807018       8.403509
2                  0.054054   15.405405       8.081081
3                  1.594059   17.643564       9.143564
```

CLUSTER CHARACTERISTICS

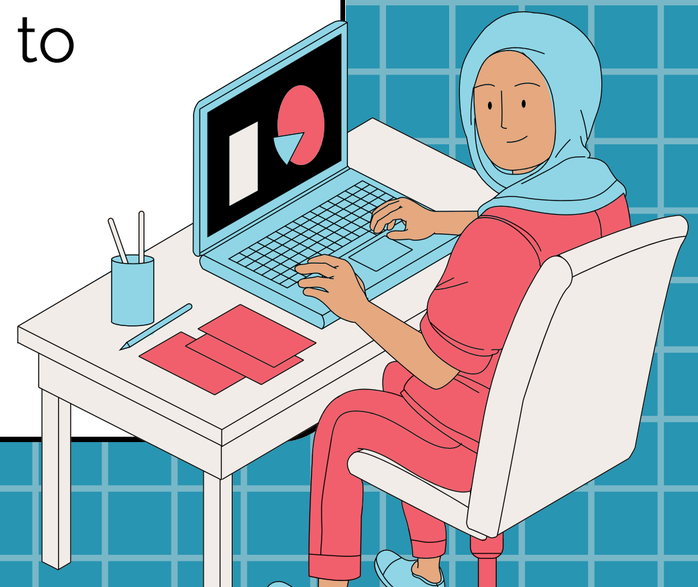


- Allows for easy comparison and identification of patterns

RESULT ANALYSIS AND FINDINGS

DATA-DRIVEN INSIGHTS

- Songs with high energy, danceability, and valence tend to perform well
- Songs released during peak seasons (summer, festivals, etc.) tend to receive more streams
- Artists with a large following and high popularity tend to have more successful songs



RECOMMENDATIONS

- Focus on creating songs with high energy, danceability, and valence
- Release songs during peak seasons
- Collaborate with popular artists or feature them in songs

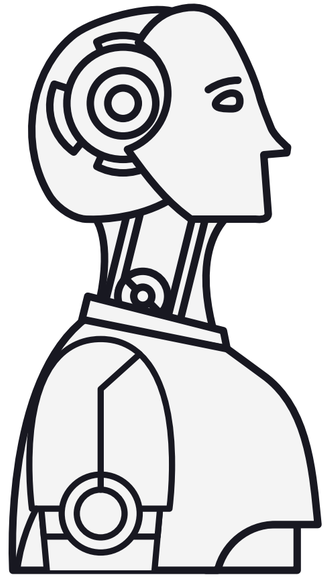
RECOMMENDED ALGORITHM

Gradient Boosting Regressor

- Suitable for predicting the number of streams a song will receive
- Handles large datasets and high-dimensional feature spaces effectively
- Can handle non-linear relationships
- Provides high accuracy and robust performance
- Can handle missing values and outliers

FUTURE

- Incorporate additional features to improve the model's accuracy
- Use deep learning models to analyse audio features and predict song success



THANK YOU!