

# Project\_4:Music\_Popularity\_Prediction\_v1

November 7, 2024

## 1 Project 4: Music Popularity Prediction

By: Robert S Balch

### 1.1 Hypothesis:

The popularity of a song on Spotify's Top 200 Weekly (Global) charts in 2020 & 2021 is likely influenced by a combination of audio features, artist popularity, and chart performance metrics. Specifically:

1. Audio Features:
  - Loudness and Energy are likely to be strong predictors of popularity, as more energetic and louder songs tend to perform better on charts.
  - Danceability and Valence (positiveness) may also be important, as upbeat and positive songs often appeal to a wider audience.
  - Tempo could be a factor, with faster-paced songs potentially being more popular in certain genres.
2. Artist Popularity:
  - The number of artist followers is likely to be a significant predictor, as more popular artists tend to have more popular songs.
3. Chart Performance Metrics:
  - Highest Charting Position and Number of Times Charted are likely to be strong indicators of overall popularity.
4. Genre:
  - Certain genres (e.g., pop, hip-hop) may be more represented in the top charts, potentially influencing popularity.
5. Song Characteristics:
  - Duration might play a role, with shorter songs potentially being more popular in recent years.
6. Release Timing:
  - The release date of the song could influence its popularity, with songs released earlier in the year potentially having more time to accumulate popularity.
7. Feature Interactions:
  - The interaction between audio features and artist popularity could be important. For example, a highly energetic song by a popular artist might be more likely to be popular than a similar song by a less known artist.
8. Cultural and Temporal Factors:
  - The dataset spans 2020 & 2021, which includes the COVID-19 pandemic period. This might have influenced listening habits and song popularity.

The [data](#). A chosen data set is provided by DDC Data Science

## 2 Imports

```
[2]: import sys
      print(sys.executable)
```

/usr/local/bin/python

```
[3]: import pandas as pd
      import numpy as np
      import matplotlib.pyplot as plt
      import matplotlib.colors as mcolors
      import seaborn as sns

      from sklearn.preprocessing import StandardScaler
      from sklearn.preprocessing import MinMaxScaler
      from sklearn.model_selection import cross_val_score
      from sklearn.model_selection import train_test_split
      from sklearn.linear_model import LinearRegression
      from sklearn.tree import DecisionTreeRegressor
      from sklearn.ensemble import RandomForestRegressor
      import xgboost as xgb

      from sklearn.metrics import mean_squared_error, root_mean_squared_error, r2_score
```

```
[4]: import pandas as pd
      import numpy as np
      import matplotlib.pyplot as plt
      import seaborn as sns
      #n_test_split
      from sklearn.linear_model import LinearRegression
      from sklearn.tree import DecisionTreeRegressor
      from sklearn.ensemble import RandomForestRegressor
      import xgboost as xgb

      from sklearn.metrics import mean_squared_error, root_mean_squared_error, r2_score
```

```
[5]: %%capture
      url = "https://ddc-datascience.s3.amazonaws.com/Projects/Project.4-Spotify/Data/
           ↳ Spotify.csv"
      !curl -s -I {url}
```

### 3 Data Exploration

```
[6]: df_1 = pd.read_csv(url).copy()
```

#### 3.1 Head

```
[7]: df_1.head()
```

```
[7]:
```

	Index	Highest Charting Position	Number of Times Charted	\
0	1	1	8	
1	2	2	3	
2	3	1	11	
3	4	3	5	
4	5	5	1	

	Week of Highest Charting	Song Name	Streams	\
0	2021-07-23--2021-07-30	Beggin'	48,633,449	
1	2021-07-23--2021-07-30	STAY (with Justin Bieber)	47,248,719	
2	2021-06-25--2021-07-02	good 4 u	40,162,559	
3	2021-07-02--2021-07-09	Bad Habits	37,799,456	
4	2021-07-23--2021-07-30	INDUSTRY BABY (feat. Jack Harlow)	33,948,454	

	Artist	Artist Followers	Song ID	\
0	Måneskin	3377762	3Wrjm47oTz2sjIgck11l5e	
1	The Kid LAROI	2230022	5HCyWlXZPP0y6Gqq8TgA20	
2	Olivia Rodrigo	6266514	4ZtFanR9U6ndgddUvNcjcG	
3	Ed Sheeran	83293380	6PQ88X9TkUIAUIZJHW2upE	
4	Lil Nas X	5473565	27NovPIUIRr0ZoCHxABJwK	

	Genre	...	Danceability	Energy	Loudness	\
0	['indie rock italiano', 'italian pop']	...	0.714	0.8	-4.808	
1	['australian hip hop']	...	0.591	0.764	-5.484	
2	['pop']	...	0.563	0.664	-5.044	
3	['pop', 'uk pop']	...	0.808	0.897	-3.712	
4	['lgbtq+ hip hop', 'pop rap']	...	0.736	0.704	-7.409	

	Speechiness	Acousticness	Liveness	Tempo	Duration (ms)	Valence	Chord
0	0.0504	0.127	0.359	134.002	211560	0.589	B
1	0.0483	0.0383	0.103	169.928	141806	0.478	C#/Db
2	0.154	0.335	0.0849	166.928	178147	0.688	A
3	0.0348	0.0469	0.364	126.026	231041	0.591	B
4	0.0615	0.0203	0.0501	149.995	212000	0.894	D#/Eb

```
[5 rows x 23 columns]
```

### 3.2 Tail

### 3.3 Shape

```
[8]: df_1.shape
```

```
[8]: (1556, 23)
```

### 3.4 columns

```
[9]: df_1.columns
```

```
[9]: Index(['Index', 'Highest Charting Position', 'Number of Times Charted',  
        'Week of Highest Charting', 'Song Name', 'Streams', 'Artist',  
        'Artist Followers', 'Song ID', 'Genre', 'Release Date', 'Weeks Charted',  
        'Popularity', 'Danceability', 'Energy', 'Loudness', 'Speechiness',  
        'Acousticness', 'Liveness', 'Tempo', 'Duration (ms)', 'Valence',  
        'Chord'],  
        dtype='object')
```

### 3.5 Dtypes

```
[10]: df_1.dtypes
```

```
[10]: Index                                int64  
Highest Charting Position                int64  
Number of Times Charted                 int64  
Week of Highest Charting                object  
Song Name                              object  
Streams                                object  
Artist                                 object  
Artist Followers                       object  
Song ID                                object  
Genre                                  object  
Release Date                           object  
Weeks Charted                          object  
Popularity                             object  
Danceability                           object  
Energy                                 object  
Loudness                               object  
Speechiness                            object  
Acousticness                           object  
Liveness                               object  
Tempo                                  object  
Duration (ms)                          object  
Valence                                object  
Chord                                   object  
dtype: object
```

### 3.6 Describe

```
[11]: df_1.describe()
```

```
[11]:
```

	Index	Highest Charting Position	Number of Times Charted
count	1556.000000	1556.000000	1556.000000
mean	778.500000	87.744216	10.668380
std	449.322824	58.147225	16.360546
min	1.000000	1.000000	1.000000
25%	389.750000	37.000000	1.000000
50%	778.500000	80.000000	4.000000
75%	1167.250000	137.000000	12.000000
max	1556.000000	200.000000	142.000000

### 3.7 Isnull Sum

```
[12]: df_1.isnull().sum()
```

```
[12]:
```

Index	0
Highest Charting Position	0
Number of Times Charted	0
Week of Highest Charting	0
Song Name	0
Streams	0
Artist	0
Artist Followers	0
Song ID	0
Genre	0
Release Date	0
Weeks Charted	0
Popularity	0
Danceability	0
Energy	0
Loudness	0
Speechiness	0
Acousticness	0
Liveness	0
Tempo	0
Duration (ms)	0
Valence	0
Chord	0
dtype: int64	

### 3.8 Isna Sum

```
[13]: df_1.isna().sum()
```

```
[13]: Index                                0
      Highest Charting Position          0
      Number of Times Charted           0
      Week of Highest Charting           0
      Song Name                          0
      Streams                           0
      Artist                            0
      Artist Followers                   0
      Song ID                           0
      Genre                             0
      Release Date                       0
      Weeks Charted                      0
      Popularity                         0
      Danceability                       0
      Energy                             0
      Loudness                           0
      Speechiness                        0
      Acousticness                       0
      Liveness                           0
      Tempo                              0
      Duration (ms)                     0
      Valence                            0
      Chord                              0
      dtype: int64
```

### 3.9 unique values

```
[14]: df_1.count('rows').unique().sum()
```

```
[14]: np.int64(1556)
```

```
[15]: df_1.count('columns')
```

```
[15]: 0      23
      1      23
      2      23
      3      23
      4      23
      ..
      1551    23
      1552    23
      1553    23
      1554    23
      1555    23
      Length: 1556, dtype: int64
```

### 3.10 Sort\_values

```
[16]: df_1.sort_values(by = ['Popularity'], ascending = False).head(10)
```

```
[16]:
```

	Index	Highest Charting Position	Number of Times Charted \
1	2	2	3
2	3	1	11
3	4	3	5
5	6	1	18
4	5	5	1
8	9	3	8
14	15	2	10
7	8	2	10
9	10	8	10
11	12	9	9

	Week of Highest Charting	Song Name	Streams \
1	2021-07-23--2021-07-30	STAY (with Justin Bieber)	47,248,719
2	2021-06-25--2021-07-02	good 4 u	40,162,559
3	2021-07-02--2021-07-09	Bad Habits	37,799,456
5	2021-05-07--2021-05-14	MONTERO (Call Me By Your Name)	30,071,134
4	2021-07-23--2021-07-30	INDUSTRY BABY (feat. Jack Harlow)	33,948,454
8	2021-06-18--2021-06-25	Yonaguni	25,030,128
14	2021-05-21--2021-05-28	Butter	19,985,713
7	2021-06-18--2021-06-25	Todo De Ti	26,951,613
9	2021-07-02--2021-07-09	I WANNA BE YOUR SLAVE	24,551,591
11	2021-07-02--2021-07-09	Qué Más Pues?	22,405,111

	Artist	Artist Followers	Song ID \
1	The Kid LAROI	2230022	5HCyWlXZPP0y6Gqq8TgA20
2	Olivia Rodrigo	6266514	4ZtFanR9U6ndgddUvNcjG
3	Ed Sheeran	83293380	6PQ88X9TkUIAUIZJHW2upE
5	Lil Nas X	5473565	67BtfxlNbhBmCDR2L2l8qd
4	Lil Nas X	5473565	27NovPIUIRr0ZoCHxABJwK
8	Bad Bunny	36142273	2JPLbjOn0wPCngEot2STUS
14	BTS	37106176	2bgTY4UwhfBYhGT4HUyStN
7	Rauw Alejandro	6080597	4fSIb4hd0Q151TILNsSEaF
9	Måneskin	3377762	4pt5fDVTg5GhEvEt1z9dKk
11	J Balvin, Maria Becerra	29051363	6hf0RpxTb0prT5nnwzkk8e

	Genre	... Danceability	Energy \
1	['australian hip hop']	...	0.591 0.764
2	['pop']	...	0.563 0.664
3	['pop', 'uk pop']	...	0.808 0.897
5	['lgbtq+ hip hop', 'pop rap']	...	0.61 0.508
4	['lgbtq+ hip hop', 'pop rap']	...	0.736 0.704
8	['latin', 'reggaeton', 'trap latino']	...	0.644 0.648

14	['k-pop', 'k-pop boy group']	...	0.759	0.459
7	['puerto rican pop', 'trap latino']	...	0.78	0.718
9	['indie rock italiano', 'italian pop']	...	0.75	0.608
11	['latin', 'reggaeton', 'reggaeton colombiano']	...	0.891	0.819

	Loudness	Speechiness	Acousticness	Liveness	Tempo	Duration (ms)	Valence \
1	-5.484	0.0483	0.0383	0.103	169.928	141806	0.478
2	-5.044	0.154	0.335	0.0849	166.928	178147	0.688
3	-3.712	0.0348	0.0469	0.364	126.026	231041	0.591
5	-6.682	0.152	0.297	0.384	178.818	137876	0.758
4	-7.409	0.0615	0.0203	0.0501	149.995	212000	0.894
8	-4.601	0.118	0.276	0.135	179.951	206710	0.44
14	-5.187	0.0948	0.00323	0.0906	109.997	164442	0.695
7	-3.605	0.0506	0.31	0.0932	127.949	199604	0.342
9	-4.008	0.0387	0.00165	0.178	132.507	173347	0.958
11	-3.964	0.106	0.0261	0.173	101.968	217773	0.768

	Chord
1	C#/Db
2	A
3	B
5	G#/Ab
4	D#/Eb
8	C#/Db
14	G#/Ab
7	D#/Eb
9	C#/Db
11	G#/Ab

[10 rows x 23 columns]

## 4 Data Cleaning and Feature Engineering

### 4.1 New copy of dataframe

```
[17]: df_cleaning = df_1.copy()
df_cleaning
```

```
[17]:
```

	Index	Highest Charting Position	Number of Times Charted \
0	1	1	8
1	2	2	3
2	3	1	11
3	4	3	5
4	5	5	1
...	...	...	...
1551	1552	195	1
1552	1553	196	1



1553	1554	197	1
1554	1555	198	1
1555	1556	199	1

	Week of Highest Charting	Song Name	Streams \
0	2021-07-23--2021-07-30	Beggin'	48,633,449
1	2021-07-23--2021-07-30	STAY (with Justin Bieber)	47,248,719
2	2021-06-25--2021-07-02	good 4 u	40,162,559
3	2021-07-02--2021-07-09	Bad Habits	37,799,456
4	2021-07-23--2021-07-30	INDUSTRY BABY (feat. Jack Harlow)	33,948,454
...	...	...	...
1551	2019-12-27--2020-01-03	New Rules	4,630,675
1552	2019-12-27--2020-01-03	Cheirosa - Ao Vivo	4,623,030
1553	2019-12-27--2020-01-03	Havana (feat. Young Thug)	4,620,876
1554	2019-12-27--2020-01-03	Surtada - Remix Brega Funk	4,607,385
1555	2019-12-27--2020-01-03	Lover (Remix) [feat. Shawn Mendes]	4,595,450

	Artist	Artist Followers	Song ID \
0	Måneskin	3377762	3Wrjm47oTz2sjIgck1115e
1	The Kid LAROI	2230022	5HCyWlXZPP0y6Gqq8TgA20
2	Olivia Rodrigo	6266514	4ZtFanR9U6ndgddUvNcjcG
3	Ed Sheeran	83293380	6PQ88X9TkUIAUJZJHW2upE
4	Lil Nas X	5473565	27NovPIUIRrOZOCHxABJwK
...	...	...	...
1551	Dua Lipa	27167675	2ekn2ttSfGqwhhate0LSR0
1552	Jorge & Mateus	15019109	2PWjKmJyTZedpmOUa3a5da
1553	Camila Cabello	22698747	1rfofaqEpACxVEHIZBJe6W
1554	Dadá Boladão, Tati Zaqui, OIK	208630	5F8ffc8KWKNaWllr5WsW0r
1555	Taylor Swift	42227614	3i9UVldZOE0aDOJnyfAZZ0

	Genre	... Danceability \
0	['indie rock italiano', 'italian pop']	... 0.714
1	['australian hip hop']	... 0.591
2	['pop']	... 0.563
3	['pop', 'uk pop']	... 0.808
4	['lgbtq+ hip hop', 'pop rap']	... 0.736
...	...	...
1551	['dance pop', 'pop', 'uk pop']	... 0.762
1552	['sertanejo', 'sertanejo universitario']	... 0.528
1553	['dance pop', 'electropop', 'pop', 'post-teen ...	... 0.765
1554	['brega funk', 'funk carioca']	... 0.832
1555	['pop', 'post-teen pop']	... 0.448

	Energy	Loudness	Speechiness	Acousticness	Liveness	Tempo	Duration (ms) \
0	0.8	-4.808	0.0504	0.127	0.359	134.002	211560
1	0.764	-5.484	0.0483	0.0383	0.103	169.928	141806
2	0.664	-5.044	0.154	0.335	0.0849	166.928	178147

3	0.897	-3.712	0.0348	0.0469	0.364	126.026	231041
4	0.704	-7.409	0.0615	0.0203	0.0501	149.995	212000
...	...	...	...	...	...	...	...
1551	0.7	-6.021	0.0694	0.00261	0.153	116.073	209320
1552	0.87	-3.123	0.0851	0.24	0.333	152.37	181930
1553	0.523	-4.333	0.03	0.184	0.132	104.988	217307
1554	0.55	-7.026	0.0587	0.249	0.182	154.064	152784
1555	0.603	-7.176	0.064	0.433	0.0862	205.272	221307

	Valence	Chord
0	0.589	B
1	0.478	C#/Db
2	0.688	A
3	0.591	B
4	0.894	D#/Eb
...	...	...
1551	0.608	A
1552	0.714	B
1553	0.394	D
1554	0.881	F
1555	0.422	G

[1556 rows x 23 columns]

## 4.2 drop Index

```
[18]: df_cleaning.drop('Index', axis = 1, inplace = True)
      #i
```

```
[19]: df_cleaning.transpose()
```

```
[19]:
```

Highest Charting Position	0	\
Number of Times Charted	1	
Week of Highest Charting	8	
Song Name	2021-07-23--2021-07-30	Beggin'
Streams	48,633,449	
Artist	Måneskin	
Artist Followers	3377762	
Song ID	3Wrjm47oTz2sjIgck11l5e	
Genre	['indie rock italiano', 'italian pop']	
Release Date	2017-12-08	
Weeks Charted	2021-07-23--2021-07-30\n2021-07-16--2021-07-23...	
Popularity	100	
Danceability	0.714	
Energy	0.8	
Loudness	-4.808	

Speechiness	0.0504
Acousticness	0.127
Liveness	0.359
Tempo	134.002
Duration (ms)	211560
Valence	0.589
Chord	B

1 \

Highest Charting Position	2
Number of Times Charted	3
Week of Highest Charting	2021-07-23--2021-07-30
Song Name	STAY (with Justin Bieber)
Streams	47,248,719
Artist	The Kid LAROI
Artist Followers	2230022
Song ID	5HCyWlXZPP0y6Gqq8TgA20
Genre	['australian hip hop']
Release Date	2021-07-09
Weeks Charted	2021-07-23--2021-07-30\n2021-07-16--2021-07-23...
Popularity	99
Danceability	0.591
Energy	0.764
Loudness	-5.484
Speechiness	0.0483
Acousticness	0.0383
Liveness	0.103
Tempo	169.928
Duration (ms)	141806
Valence	0.478
Chord	C#/Db

2 \

Highest Charting Position	1
Number of Times Charted	11
Week of Highest Charting	2021-06-25--2021-07-02
Song Name	good 4 u
Streams	40,162,559
Artist	Olivia Rodrigo
Artist Followers	6266514
Song ID	4ZtFanR9U6ndgddUvNcjcG
Genre	['pop']
Release Date	2021-05-21
Weeks Charted	2021-07-23--2021-07-30\n2021-07-16--2021-07-23...
Popularity	99
Danceability	0.563
Energy	0.664

Loudness	-5.044
Speechiness	0.154
Acousticness	0.335
Liveness	0.0849
Tempo	166.928
Duration (ms)	178147
Valence	0.688
Chord	A

3 \

Highest Charting Position	3
Number of Times Charted	5
Week of Highest Charting	2021-07-02--2021-07-09
Song Name	Bad Habits
Streams	37,799,456
Artist	Ed Sheeran
Artist Followers	83293380
Song ID	6PQ88X9TkUIAUIZJHW2upE
Genre	['pop', 'uk pop']
Release Date	2021-06-25
Weeks Charted	2021-07-23--2021-07-30\n2021-07-16--2021-07-23...
Popularity	98
Danceability	0.808
Energy	0.897
Loudness	-3.712
Speechiness	0.0348
Acousticness	0.0469
Liveness	0.364
Tempo	126.026
Duration (ms)	231041
Valence	0.591
Chord	B

4 \

Highest Charting Position	5
Number of Times Charted	1
Week of Highest Charting	2021-07-23--2021-07-30
Song Name	INDUSTRY BABY (feat. Jack Harlow)
Streams	33,948,454
Artist	Lil Nas X
Artist Followers	5473565
Song ID	27NovPIUIRr0ZoCHxABJwK
Genre	['lgbtq+ hip hop', 'pop rap']
Release Date	2021-07-23
Weeks Charted	2021-07-23--2021-07-30
Popularity	96
Danceability	0.736

Energy	0.704
Loudness	-7.409
Speechiness	0.0615
Acousticness	0.0203
Liveness	0.0501
Tempo	149.995
Duration (ms)	212000
Valence	0.894
Chord	D#/Eb

	5	\
Highest Charting Position	1	
Number of Times Charted	18	
Week of Highest Charting	2021-05-07--2021-05-14	
Song Name	MONTERO (Call Me By Your Name)	
Streams	30,071,134	
Artist	Lil Nas X	
Artist Followers	5473565	
Song ID	67BtfxlNbBmCDR2L2l8qd	
Genre	['lgbtq+ hip hop', 'pop rap']	
Release Date	2021-03-31	
Weeks Charted	2021-07-23--2021-07-30\n2021-07-16--2021-07-23...	
Popularity	97	
Danceability	0.61	
Energy	0.508	
Loudness	-6.682	
Speechiness	0.152	
Acousticness	0.297	
Liveness	0.384	
Tempo	178.818	
Duration (ms)	137876	
Valence	0.758	
Chord	G#/Ab	

	6	\
Highest Charting Position	3	
Number of Times Charted	16	
Week of Highest Charting	2021-05-14--2021-05-21	
Song Name	Kiss Me More (feat. SZA)	
Streams	29,356,736	
Artist	Doja Cat	
Artist Followers	8640063	
Song ID	748mdHapucXQri7IA08yFK	
Genre	['dance pop', 'pop']	
Release Date	2021-04-09	
Weeks Charted	2021-07-23--2021-07-30\n2021-07-16--2021-07-23...	
Popularity	94	

Danceability	0.762
Energy	0.701
Loudness	-3.541
Speechiness	0.0286
Acousticness	0.235
Liveness	0.123
Tempo	110.968
Duration (ms)	208867
Valence	0.742
Chord	G#/Ab

	7	\
Highest Charting Position	2	
Number of Times Charted	10	
Week of Highest Charting	2021-06-18--2021-06-25	
Song Name	Todo De Ti	
Streams	26,951,613	
Artist	Rauw Alejandro	
Artist Followers	6080597	
Song ID	4fSIb4hd0Q151TILNsSEaF	
Genre	['puerto rican pop', 'trap latino']	
Release Date	2021-05-20	
Weeks Charted	2021-07-23--2021-07-30\n2021-07-16--2021-07-23...	
Popularity	95	
Danceability	0.78	
Energy	0.718	
Loudness	-3.605	
Speechiness	0.0506	
Acousticness	0.31	
Liveness	0.0932	
Tempo	127.949	
Duration (ms)	199604	
Valence	0.342	
Chord	D#/Eb	

	8	\
Highest Charting Position	3	
Number of Times Charted	8	
Week of Highest Charting	2021-06-18--2021-06-25	
Song Name	Yonaguni	
Streams	25,030,128	
Artist	Bad Bunny	
Artist Followers	36142273	
Song ID	2JPLbj0n0wPCngEot2STUS	
Genre	['latin', 'reggaeton', 'trap latino']	
Release Date	2021-06-04	
Weeks Charted	2021-07-23--2021-07-30\n2021-07-16--2021-07-23...	

Popularity	96
Danceability	0.644
Energy	0.648
Loudness	-4.601
Speechiness	0.118
Acousticness	0.276
Liveness	0.135
Tempo	179.951
Duration (ms)	206710
Valence	0.44
Chord	C#/Db

	9	\
Highest Charting Position	8	
Number of Times Charted	10	
Week of Highest Charting	2021-07-02--2021-07-09	
Song Name	I WANNA BE YOUR SLAVE	
Streams	24,551,591	
Artist	Måneskin	
Artist Followers	3377762	
Song ID	4pt5fDVTg5GhEvEtlz9dKk	
Genre	['indie rock italiano', 'italian pop']	
Release Date	2021-03-19	
Weeks Charted	2021-07-23--2021-07-30\n2021-07-16--2021-07-23...	
Popularity	95	
Danceability	0.75	
Energy	0.608	
Loudness	-4.008	
Speechiness	0.0387	
Acousticness	0.00165	
Liveness	0.178	
Tempo	132.507	
Duration (ms)	173347	
Valence	0.958	
Chord	C#/Db	

	...	1546	\
Highest Charting Position	...	143	
Number of Times Charted	...	1	
Week of Highest Charting	...	2019-12-27--2020-01-03	
Song Name	...	JACKBOYS	
Streams	...	5,363,493	
Artist	...	JACKBOYS	
Artist Followers	...	437907	
Song ID	...	62zKJrpbLxz6InR3tGyr7o	
Genre	...	['rap', 'trap']	
Release Date	...	2019-12-27	

Weeks Charted	...	2019-12-27--2020-01-03
Popularity	...	56
Danceability	...	0.413
Energy	...	0.13
Loudness	...	-25.166
Speechiness	...	0.0336
Acousticness	...	0.9
Liveness	...	0.111
Tempo	...	123.342
Duration (ms)	...	46837
Valence	...	0.0676
Chord	...	C

		1547 \
Highest Charting Position		156
Number of Times Charted		1
Week of Highest Charting		2019-12-27--2020-01-03
Song Name		Combatchy (feat. MC Rebecca)
Streams		5,149,797
Artist		Anitta, Lexa, Luísa Sonza
Artist Followers		10741972
Song ID		2bPtwnrpFNEe8N7Q85kLHw
Genre		['funk carioca', 'funk pop', 'pagode baiano', ...]
Release Date		2019-11-20
Weeks Charted		2019-12-27--2020-01-03
Popularity		64
Danceability		0.826
Energy		0.73
Loudness		-3.032
Speechiness		0.0809
Acousticness		0.383
Liveness		0.0197
Tempo		150.134
Duration (ms)		157600
Valence		0.605
Chord		C#/Db

		1548 \
Highest Charting Position		178
Number of Times Charted		1
Week of Highest Charting		2019-12-27--2020-01-03
Song Name		Old Town Road
Streams		4,852,004
Artist		Lil Nas X
Artist Followers		5488666
Song ID		2YpeDb67231RjR0MgVLzsG
Genre		['lgbtq+ hip hop', 'pop rap']



Release Date	2019-06-21
Weeks Charted	2019-12-27--2020-01-03
Popularity	81
Danceability	0.878
Energy	0.619
Loudness	-5.56
Speechiness	0.102
Acousticness	0.0533
Liveness	0.113
Tempo	136.041
Duration (ms)	157067
Valence	0.639
Chord	F#/Gb

	1549 \
Highest Charting Position	187
Number of Times Charted	1
Week of Highest Charting	2019-12-27--2020-01-03
Song Name	Let Me Know (I Wonder Why Freestyle)
Streams	4,701,532
Artist	Juice WRLD
Artist Followers	19102888
Song ID	3wwo0bJvDSorOpNfzEkfXx
Genre	['chicago rap', 'melodic rap']
Release Date	2019-12-07
Weeks Charted	2019-12-27--2020-01-03
Popularity	76
Danceability	0.635
Energy	0.537
Loudness	-7.895
Speechiness	0.0832
Acousticness	0.172
Liveness	0.418
Tempo	125.028
Duration (ms)	215381
Valence	0.383
Chord	G

	1550 \
Highest Charting Position	190
Number of Times Charted	1
Week of Highest Charting	2019-12-27--2020-01-03
Song Name	Ne reviens pas
Streams	4,676,857
Artist	Gradur, Heuss L'enfoiré
Artist Followers	1390813
Song ID	4TnFANpjVwVKWzKxNzIyFH

Genre	['francoton', 'french hip hop', 'pop urbaine',...
Release Date	2019-11-29
Weeks Charted	2019-12-27--2020-01-03
Popularity	62
Danceability	0.932
Energy	0.778
Loudness	-3.384
Speechiness	0.0638
Acousticness	0.212
Liveness	0.168
Tempo	124.996
Duration (ms)	188613
Valence	0.933
Chord	A#/Bb

	1551 \
Highest Charting Position	195
Number of Times Charted	1
Week of Highest Charting	2019-12-27--2020-01-03
Song Name	New Rules
Streams	4,630,675
Artist	Dua Lipa
Artist Followers	27167675
Song ID	2ekn2ttSfGqwhhate0LSR0
Genre	['dance pop', 'pop', 'uk pop']
Release Date	2017-06-02
Weeks Charted	2019-12-27--2020-01-03
Popularity	79
Danceability	0.762
Energy	0.7
Loudness	-6.021
Speechiness	0.0694
Acousticness	0.00261
Liveness	0.153
Tempo	116.073
Duration (ms)	209320
Valence	0.608
Chord	A

	1552 \
Highest Charting Position	196
Number of Times Charted	1
Week of Highest Charting	2019-12-27--2020-01-03
Song Name	Cheirosa - Ao Vivo
Streams	4,623,030
Artist	Jorge & Mateus
Artist Followers	15019109

Song ID	2PWjKmjyTZeDpmOUa3a5da
Genre	['sertanejo', 'sertanejo universitario']
Release Date	2019-10-11
Weeks Charted	2019-12-27--2020-01-03
Popularity	66
Danceability	0.528
Energy	0.87
Loudness	-3.123
Speechiness	0.0851
Acousticness	0.24
Liveness	0.333
Tempo	152.37
Duration (ms)	181930
Valence	0.714
Chord	B

	1553 \
Highest Charting Position	197
Number of Times Charted	1
Week of Highest Charting	2019-12-27--2020-01-03
Song Name	Havana (feat. Young Thug)
Streams	4,620,876
Artist	Camila Cabello
Artist Followers	22698747
Song ID	1rfofaqEpACxVEHIZBJe6W
Genre	['dance pop', 'electropop', 'pop', 'post-teen ...
Release Date	2018-01-12
Weeks Charted	2019-12-27--2020-01-03
Popularity	81
Danceability	0.765
Energy	0.523
Loudness	-4.333
Speechiness	0.03
Acousticness	0.184
Liveness	0.132
Tempo	104.988
Duration (ms)	217307
Valence	0.394
Chord	D

	1554 \
Highest Charting Position	198
Number of Times Charted	1
Week of Highest Charting	2019-12-27--2020-01-03
Song Name	Surtada - Remix Brega Funk
Streams	4,607,385
Artist	Dadá Boladão, Tati Zaqui, OIK

Artist Followers	208630
Song ID	5F8ffc8KWKNawllr5WsW0r
Genre	['brega funk', 'funk carioca']
Release Date	2019-09-25
Weeks Charted	2019-12-27--2020-01-03
Popularity	60
Danceability	0.832
Energy	0.55
Loudness	-7.026
Speechiness	0.0587
Acousticness	0.249
Liveness	0.182
Tempo	154.064
Duration (ms)	152784
Valence	0.881
Chord	F
	1555
Highest Charting Position	199
Number of Times Charted	1
Week of Highest Charting	2019-12-27--2020-01-03
Song Name	Lover (Remix) [feat. Shawn Mendes]
Streams	4,595,450
Artist	Taylor Swift
Artist Followers	42227614
Song ID	3i9UVldZ0E0aD0JnyfAZZ0
Genre	['pop', 'post-teen pop']
Release Date	2019-11-13
Weeks Charted	2019-12-27--2020-01-03
Popularity	70
Danceability	0.448
Energy	0.603
Loudness	-7.176
Speechiness	0.064
Acousticness	0.433
Liveness	0.0862
Tempo	205.272
Duration (ms)	221307
Valence	0.422
Chord	G

[22 rows x 1556 columns]

### 4.3 Convert object columns with numbers to float64

```
[20]: # List of columns to convert
columns_to_convert = ['Artist Followers', 'Streams', 'Popularity',
    ↳ 'Danceability', 'Energy', 'Loudness',
    ↳ 'Speechiness', 'Acousticness', 'Liveness', 'Tempo',
    ↳ 'Duration (ms)', 'Valence']

df_1[columns_to_convert] = df_1[columns_to_convert].apply(pd.to_numeric,
    ↳ errors='coerce')
```

```
[21]: df_1.dtypes
```

```
[21]: Index                                int64
Highest Charting Position              int64
Number of Times Charted                int64
Week of Highest Charting               object
Song Name                             object
Streams                               float64
Artist                                object
Artist Followers                       float64
Song ID                               object
Genre                                 object
Release Date                          object
Weeks Charted                         object
Popularity                            float64
Danceability                           float64
Energy                                float64
Loudness                              float64
Speechiness                           float64
Acousticness                          float64
Liveness                              float64
Tempo                                 float64
Duration (ms)                         float64
Valence                               float64
Chord                                  object
dtype: object
```

## 5 Data Cleaning Continued: Prepare DataFrame for Modeling and Training

```
[22]: df_1 = df_1.drop("Index", axis = 1)
```

```
[23]: df_1
```

[23]:	Highest Charting Position	Number of Times Charted	\	
0		1	8	
1		2	3	
2		1	11	
3		3	5	
4		5	1	
...	...	...		
1551	195	1		
1552	196	1		
1553	197	1		
1554	198	1		
1555	199	1		
	Week of Highest Charting	Song Name	Streams	\
0	2021-07-23--2021-07-30	Beggin'	NaN	
1	2021-07-23--2021-07-30	STAY (with Justin Bieber)	NaN	
2	2021-06-25--2021-07-02	good 4 u	NaN	
3	2021-07-02--2021-07-09	Bad Habits	NaN	
4	2021-07-23--2021-07-30	INDUSTRY BABY (feat. Jack Harlow)	NaN	
...	...	...	...	
1551	2019-12-27--2020-01-03	New Rules	NaN	
1552	2019-12-27--2020-01-03	Cheirosa - Ao Vivo	NaN	
1553	2019-12-27--2020-01-03	Havana (feat. Young Thug)	NaN	
1554	2019-12-27--2020-01-03	Surtada - Remix Brega Funk	NaN	
1555	2019-12-27--2020-01-03	Lover (Remix) [feat. Shawn Mendes]	NaN	
	Artist	Artist Followers	Song ID	\
0	Måneskin	3377762.0	3Wrjm47oTz2sjIgck11l5e	
1	The Kid LAROI	2230022.0	5HCyWlXZPP0y6Gqq8TgA20	
2	Olivia Rodrigo	6266514.0	4ZtFanR9U6ndgddUvNcjcG	
3	Ed Sheeran	83293380.0	6PQ88X9TkUIAUIZJHW2upE	
4	Lil Nas X	5473565.0	27NovPIUIRrOZOCHxABJwK	
...	...	...	...	
1551	Dua Lipa	27167675.0	2ekn2ttSfGqwhhate0LSR0	
1552	Jorge & Mateus	15019109.0	2PWjKmjtZeDpmOUa3a5da	
1553	Camila Cabello	22698747.0	1rfofaqEpACxVEHIZBJe6W	
1554	Dadá Boladão, Tati Zaqui, OIK	208630.0	5F8ffc8KWKNaWllr5WsW0r	
1555	Taylor Swift	42227614.0	3i9UVldZOE0aD0JnyfAZZ0	
	Genre	Release Date	...	\
0	['indie rock italiano', 'italian pop']	2017-12-08	...	
1	['australian hip hop']	2021-07-09	...	
2	['pop']	2021-05-21	...	
3	['pop', 'uk pop']	2021-06-25	...	
4	['lgbtq+ hip hop', 'pop rap']	2021-07-23	...	
...	...	...	...	
1551	['dance pop', 'pop', 'uk pop']	2017-06-02	...	

```

1552          ['sertanejo', 'sertanejo universitario'] 2019-10-11 ...
1553 ['dance pop', 'electropop', 'pop', 'post-teen ... 2018-01-12 ...
1554          ['brega funk', 'funk carioca'] 2019-09-25 ...
1555          ['pop', 'post-teen pop'] 2019-11-13 ...

```

	Danceability	Energy	Loudness	Speechiness	Acousticness	Liveness	\
0	0.714	0.800	-4.808	0.0504	0.12700	0.3590	
1	0.591	0.764	-5.484	0.0483	0.03830	0.1030	
2	0.563	0.664	-5.044	0.1540	0.33500	0.0849	
3	0.808	0.897	-3.712	0.0348	0.04690	0.3640	
4	0.736	0.704	-7.409	0.0615	0.02030	0.0501	
...	...	...	...	...	...	...	
1551	0.762	0.700	-6.021	0.0694	0.00261	0.1530	
1552	0.528	0.870	-3.123	0.0851	0.24000	0.3330	
1553	0.765	0.523	-4.333	0.0300	0.18400	0.1320	
1554	0.832	0.550	-7.026	0.0587	0.24900	0.1820	
1555	0.448	0.603	-7.176	0.0640	0.43300	0.0862	

	Tempo	Duration (ms)	Valence	Chord
0	134.002	211560.0	0.589	B
1	169.928	141806.0	0.478	C#/Db
2	166.928	178147.0	0.688	A
3	126.026	231041.0	0.591	B
4	149.995	212000.0	0.894	D#/Eb
...	...	...	...	...
1551	116.073	209320.0	0.608	A
1552	152.370	181930.0	0.714	B
1553	104.988	217307.0	0.394	D
1554	154.064	152784.0	0.881	F
1555	205.272	221307.0	0.422	G

[1556 rows x 22 columns]

```
[24]: df_clean_2 = df_1.copy()
```

## 5.1 Identify Object Columns & Drop them

```
[25]: object_columns = df_clean_2.select_dtypes(include=['object']).columns
df_clean_2 = df_clean_2.drop(columns=object_columns)
```

```
[26]: df_clean_2.isnull().sum()
```

```
[26]: Highest Charting Position      0
Number of Times Charted             0
Streams                           1556
Artist Followers                   11
Popularity                         11
```

```

Danceability      11
Energy            11
Loudness          11
Speechiness       11
Acousticness      11
Liveness          11
Tempo            11
Duration (ms)     11
Valence           11
dtype: int64

```

```
[27]: df_clean_2.info()
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1556 entries, 0 to 1555
Data columns (total 14 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Highest Charting Position             1556 non-null   int64
1   Number of Times Charted               1556 non-null   int64
2   Streams                              0 non-null      float64
3   Artist Followers                     1545 non-null   float64
4   Popularity                           1545 non-null   float64
5   Danceability                         1545 non-null   float64
6   Energy                               1545 non-null   float64
7   Loudness                             1545 non-null   float64
8   Speechiness                          1545 non-null   float64
9   Acousticness                         1545 non-null   float64
10  Liveness                             1545 non-null   float64
11  Tempo                                1545 non-null   float64
12  Duration (ms)                        1545 non-null   float64
13  Valence                              1545 non-null   float64
dtypes: float64(12), int64(2)
memory usage: 170.3 KB

```

## 5.2 Drop Streams Column (essentially empty)

```
[28]: df_clean_2.drop('Streams', axis = 1, inplace = True)
```

```
[29]: df_clean_2.info()
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1556 entries, 0 to 1555
Data columns (total 13 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Highest Charting Position             1556 non-null   int64
1   Number of Times Charted               1556 non-null   int64

```



2	Artist Followers	1545	non-null	float64
3	Popularity	1545	non-null	float64
4	Danceability	1545	non-null	float64
5	Energy	1545	non-null	float64
6	Loudness	1545	non-null	float64
7	Speechiness	1545	non-null	float64
8	Acousticness	1545	non-null	float64
9	Liveness	1545	non-null	float64
10	Tempo	1545	non-null	float64
11	Duration (ms)	1545	non-null	float64
12	Valence	1545	non-null	float64

dtypes: float64(11), int64(2)  
memory usage: 158.2 KB

### 5.3 Get means and replace null values with mean per column

```
[30]: df_clean_2.isna().sum()
```

```
[30]: Highest Charting Position      0
      Number of Times Charted      0
      Artist Followers             11
      Popularity                   11
      Danceability                  11
      Energy                       11
      Loudness                     11
      Speechiness                   11
      Acousticness                  11
      Liveness                      11
      Tempo                        11
      Duration (ms)                 11
      Valence                       11
      dtype: int64
```

```
[31]: null_columns = df_clean_2.columns[df_clean_2.isnull().any()].tolist()
      print("Columns with null values:")
      null_columns
```

Columns with null values:

```
[31]: ['Artist Followers',
      'Popularity',
      'Danceability',
      'Energy',
      'Loudness',
      'Speechiness',
      'Acousticness',
      'Liveness',
      'Tempo',
```

```
'Duration (ms)',  
'Valence']
```

```
[32]: for col in null_columns:  
        #Calculate the mean, excluding NaN values  
        mean= df_clean_2[col].mean(skipna=True)  
  
        #replace NaNs with the mean per column  
        df_clean_2[col] = df_clean_2[col].fillna(mean)
```

```
[33]: print("\nNull value count after replacement:")  
        print(df_clean_2.isnull().sum())
```

Null value count after replacement:

Highest Charting Position	0
Number of Times Charted	0
Artist Followers	0
Popularity	0
Danceability	0
Energy	0
Loudness	0
Speechiness	0
Acousticness	0
Liveness	0
Tempo	0
Duration (ms)	0
Valence	0
dtype:	int64

```
[34]: df_clean_2.dtypes
```

```
[34]: Highest Charting Position      int64  
       Number of Times Charted      int64  
       Artist Followers             float64  
       Popularity                   float64  
       Danceability                 float64  
       Energy                       float64  
       Loudness                     float64  
       Speechiness                  float64  
       Acousticness                 float64  
       Liveness                     float64  
       Tempo                       float64  
       Duration (ms)               float64  
       Valence                      float64  
       dtype: object
```

## 5.4 Drop columns that have no relation to target = “Popularity”

```
[35]: # df_clean_2.drop('Highest Charting Position', axis = 1, inplace = True)
```

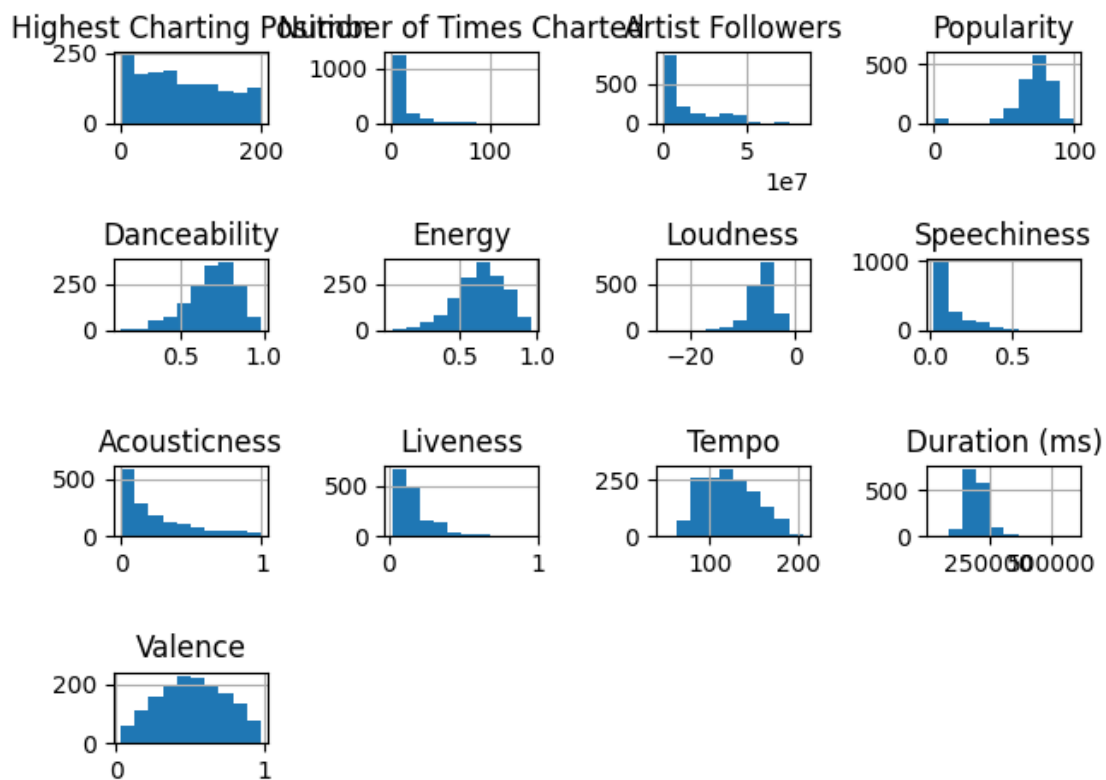
```
[36]: # df_clean_2.drop('Number of Times Charted', axis = 1, inplace = True)
```

```
[37]: # df_clean_2.drop('Artist Followers', axis = 1, inplace = True)
```

```
[38]: df_scaling = df_clean_2.copy()
```

```
[39]: df_scaling.hist()  
plt.tight_layout()  
plt.show
```

```
[39]: <function matplotlib.pyplot.show(close=None, block=None)>
```



## 6 Data Scaling

### 6.1 Data Scaling (standard scaler)

#### 6.1.1 Setup standard scaled training and testing data

```
[40]: df_3_std = df_scaling.copy()

[41]: x1 = df_3_std.drop(['Popularity'], axis=1)
      y1 = df_3_std['Popularity']

      X_train_1, X_test_1, y_train_1, y_test_1 = train_test_split(x1, y1, test_size=0.
      ↪2)

[42]: scaler = StandardScaler()
      X_train_std = scaler.fit_transform(X_train_1)
      X_test_std = scaler.transform(X_test_1)

[43]: print("Before scaling:")
      print(X_train_1.describe())

      print("\nAfter scaling:")
      print(pd.DataFrame(X_train_std).describe())
```

Before scaling:

	Highest Charting Position	Number of Times Charted	Artist Followers	\
count	1244.000000	1244.000000	1.244000e+03	
mean	87.378617	10.730707	1.490597e+07	
std	58.376581	16.433850	1.668620e+07	
min	1.000000	1.000000	4.883000e+03	
25%	36.750000	1.000000	2.203386e+06	
50%	79.000000	4.000000	6.874642e+06	
75%	137.000000	12.000000	2.298618e+07	
max	200.000000	142.000000	8.333778e+07	

	Danceability	Energy	Loudness	Speechiness	Acousticness	\
count	1244.000000	1244.000000	1244.000000	1244.000000	1244.000000	
mean	0.690860	0.632947	-6.389895	0.123600	0.248298	
std	0.141665	0.163021	2.527550	0.109979	0.248737	
min	0.150000	0.054000	-25.166000	0.025300	0.000038	
25%	0.599000	0.532000	-7.590500	0.045775	0.048225	
50%	0.708000	0.642000	-6.057500	0.077800	0.160000	
75%	0.797000	0.753000	-4.725000	0.164000	0.387250	
max	0.980000	0.970000	1.509000	0.884000	0.979000	

	Liveness	Tempo	Duration (ms)	Valence
count	1244.000000	1244.000000	1244.000000	1244.000000
mean	0.181037	122.806405	197995.867793	0.514345
std	0.144104	29.280576	46934.318670	0.227952

min	0.019700	64.062000	30133.000000	0.032000
25%	0.096525	98.021250	169222.000000	0.342750
50%	0.124000	122.205000	194159.000000	0.514704
75%	0.216250	143.802250	218640.750000	0.691250
max	0.962000	205.272000	588139.000000	0.979000

After scaling:

	0	1	2	3	4 \
count	1.244000e+03	1.244000e+03	1.244000e+03	1.244000e+03	1.244000e+03
mean	4.854995e-17	-3.284261e-17	5.711758e-18	1.713527e-16	-7.967903e-16
std	1.000402e+00	1.000402e+00	1.000402e+00	1.000402e+00	1.000402e+00
min	-1.480274e+00	-5.923518e-01	-8.933773e-01	-3.819422e+00	-3.552783e+00
25%	-8.676250e-01	-5.923518e-01	-7.615685e-01	-6.486936e-01	-6.194739e-01
50%	-1.435848e-01	-4.097284e-01	-4.815087e-01	1.210377e-01	5.555544e-02
75%	8.503639e-01	7.726753e-02	4.844397e-01	7.495340e-01	7.367214e-01
max	1.929998e+00	7.990951e+00	4.102751e+00	2.041835e+00	2.068370e+00

	5	6	7	8	9 \
count	1.244000e+03	1.244000e+03	1.244000e+03	1.244000e+03	1.244000e+03
mean	1.228028e-16	-8.853225e-17	2.141909e-16	-1.085234e-16	-2.684526e-16
std	1.000402e+00	1.000402e+00	1.000402e+00	1.000402e+00	1.000402e+00
min	-7.431567e+00	-8.941640e-01	-9.984821e-01	-1.120033e+00	-2.007065e+00
25%	-4.751983e-01	-7.079169e-01	-8.046794e-01	-5.866979e-01	-8.468113e-01
50%	1.315618e-01	-4.166073e-01	-3.551292e-01	-3.959607e-01	-2.054765e-02
75%	6.589642e-01	3.674953e-01	5.588524e-01	2.444584e-01	7.173455e-01
max	3.126376e+00	6.916844e+00	2.938824e+00	5.421613e+00	2.817525e+00

	10	11
count	1.244000e+03	1.244000e+03
mean	1.906299e-16	3.926834e-17
std	1.000402e+00	1.000402e+00
min	-3.577987e+00	-2.116839e+00
25%	-6.133133e-01	-7.530677e-01
50%	-8.178261e-02	1.576313e-03
75%	4.400444e-01	7.763745e-01
max	8.315877e+00	2.039207e+00

```
[44]: print("Mean:", X_train_std.mean(axis=0))
      print("Std:", X_train_std.std(axis=0))
```

```
Mean: [ 4.85499458e-17 -3.28426104e-17  5.71175833e-18  1.71352750e-16
        -7.96790286e-16  1.22802804e-16 -8.85322541e-17  2.14190937e-16
        -1.08523408e-16 -2.68452641e-16  1.90629934e-16  3.92683385e-17]
Std: [1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.]
```

## 6.2 Data Scaling Continued (min-max scaler)

```
[45]: df_3_mm = df_scaling.copy()
```

```
[46]: x2 = df_3_mm.drop(['Popularity'], axis=1)
      y2 = df_3_mm['Popularity']

      X_train_2, X_test_2, y_train_2, y_test_2 = train_test_split(x2, y2, test_size=0.
      ↪2)
```

### 6.2.1 Setup mm scaled training and testing data

```
[47]: scaler = MinMaxScaler()
      X_train_mm = scaler.fit_transform(X_train_2)
      X_test_mm = scaler.transform(X_test_2)
```

```
[48]: print("Before scaling:")
      print(X_train_2.describe())

      print("\nAfter scaling:")
      print(pd.DataFrame(X_train_mm).describe())
```

Before scaling:

	Highest Charting Position	Number of Times Charted	Artist Followers	\
count	1244.000000	1244.000000	1.244000e+03	
mean	88.681672	10.509646	1.501797e+07	
std	58.507613	16.553744	1.684511e+07	
min	1.000000	1.000000	4.883000e+03	
25%	38.000000	1.000000	2.158176e+06	
50%	80.000000	3.000000	7.358720e+06	
75%	138.000000	12.000000	2.384846e+07	
max	200.000000	142.000000	8.333778e+07	

	Danceability	Energy	Loudness	Speechiness	Acousticness	\
count	1244.000000	1244.000000	1244.000000	1244.000000	1244.000000	
mean	0.691579	0.629196	-6.381143	0.121322	0.250824	
std	0.140081	0.161058	2.520145	0.108095	0.251625	
min	0.218000	0.054000	-25.166000	0.023200	0.000038	
25%	0.604000	0.527750	-7.502500	0.045775	0.047125	
50%	0.708000	0.640000	-6.002500	0.075450	0.165000	
75%	0.795000	0.746250	-4.732500	0.160000	0.390000	
max	0.980000	0.970000	1.509000	0.884000	0.994000	

	Liveness	Tempo	Duration (ms)	Valence
count	1244.000000	1244.000000	1244.000000	1244.000000
mean	0.179280	122.169576	198110.222442	0.515052
std	0.142483	29.263083	47608.846179	0.227942
min	0.027300	46.718000	30133.000000	0.032000

25%	0.097050	97.006500	168889.500000	0.342750
50%	0.123000	121.955000	193843.000000	0.514704
75%	0.210000	142.768000	218162.750000	0.695000
max	0.962000	205.272000	588139.000000	0.977000

After scaling:

	0	1	2	3	4 \
count	1244.000000	1244.000000	1244.000000	1244.000000	1244.000000
mean	0.440611	0.067444	0.180158	0.621494	0.627943
std	0.294008	0.117402	0.202142	0.183833	0.175828
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.185930	0.000000	0.025840	0.506562	0.517194
50%	0.396985	0.014184	0.088247	0.643045	0.639738
75%	0.688442	0.078014	0.286124	0.757218	0.755731
max	1.000000	1.000000	1.000000	1.000000	1.000000

	5	6	7	8	9 \
count	1244.000000	1244.000000	1244.000000	1244.000000	1244.000000
mean	0.704212	0.113989	0.252309	0.162597	0.475873
std	0.094476	0.125575	0.253154	0.152437	0.184562
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.662174	0.026226	0.047373	0.074623	0.317170
50%	0.718407	0.060699	0.165964	0.102386	0.474520
75%	0.766017	0.158922	0.392331	0.195464	0.605787
max	1.000000	1.000000	1.000000	1.000000	1.000000

	10	11
count	1244.000000	1244.000000
mean	0.301031	0.511166
std	0.085320	0.241209
min	0.000000	0.000000
25%	0.248665	0.328836
50%	0.293384	0.510798
75%	0.336967	0.701587
max	1.000000	1.000000

```
[49]: print("Mean:", X_train_mm.mean(axis=0))
      print("Std:", X_train_mm.std(axis=0))
```

```
Mean: [0.44061142 0.0674443 0.18015796 0.62149444 0.62794292 0.70421206
0.11398924 0.25230869 0.16259711 0.47587306 0.30103121 0.511166 ]
Std: [0.29388991 0.11735525 0.20206115 0.18375903 0.17575701 0.09443794
0.12552476 0.25305236 0.1523759 0.18448806 0.0852853 0.24111157]
```

## 7 Model Selection and Training

### 7.1 Models: STD Scaler

#### 7.1.1 Linear Regression std scaler

```
[50]: lr_model = LinearRegression()
lr_model.fit(X_train_std, y_train_1)
y_pred_lr = lr_model.predict(X_test_std)
print('Linear Regression:')
print(f"RMSE: {np.sqrt(mean_squared_error(y_test_1,y_pred_lr)) :.2f}%")
print(f"R2 Score: {r2_score(y_test_1,y_pred_lr):.2f}")
```

Linear Regression:

RMSE: 13.66%

R2 Score: 0.11

#### Cross Validation Score for Linear Regression

```
[51]: lr_model = LinearRegression()
cv_scores = cross_val_score(lr_model, X_train_1, y_train_1, cv=5,
    ↳scoring='neg_mean_squared_error')
rmse = np.sqrt(-cv_scores.mean())
print(f"Cross-validated RMSE: {rmse:.2f}")
```

Cross-validated RMSE: 15.52

#### 7.1.2 Decision Tree Model std scaler

```
[52]: dt_model = DecisionTreeRegressor()
dt_model.fit(X_train_std, y_train_1)
y_pred_dt = dt_model.predict(X_test_std)

print("\nDecision Tree:")
print(f"RMSE: {np.sqrt(mean_squared_error(y_test_1, y_pred_dt)) :.2f}%")
print(f"R2 Score: {r2_score(y_test_1, y_pred_dt):.2f}")
```

Decision Tree:

RMSE: 11.60%

R2 Score: 0.36

#### Cross Validation Score for Decision Tree

```
[53]: dt_model = DecisionTreeRegressor()
cv_scores = cross_val_score(dt_model, X_train_std, y_train_1, cv=5,
    ↳scoring='neg_mean_squared_error')
rmse = np.sqrt(-cv_scores.mean())
print(f"Cross-validated RMSE: {rmse:.2f}")
```

Cross-validated RMSE: 11.77



### Feature Importance for Decision Tree

```
[54]: dt_model.fit(X_train_std, y_train_1)

feature_importances = dt_model.feature_importances_
feature_names = X_train_1.columns
feature_importance_df = pd.DataFrame({'feature': feature_names, 'importance':
    ↳feature_importances})
feature_importance_df = feature_importance_df.sort_values('importance',
    ↳ascending=False)
print(feature_importance_df)
```

	feature	importance
2	Artist Followers	0.622208
1	Number of Times Charted	0.112418
0	Highest Charting Position	0.054779
5	Loudness	0.034382
3	Danceability	0.033624
8	Liveness	0.026943
4	Energy	0.022914
11	Valence	0.022900
6	Speechiness	0.021803
7	Acousticness	0.020372
10	Duration (ms)	0.015179
9	Tempo	0.012478

### 7.1.3 Random Forest Model std scaler

```
[55]: rf_model = RandomForestRegressor(n_estimators=100)
rf_model.fit(X_train_std, y_train_1)
y_pred_rf = rf_model.predict(X_test_std)

print("\nRandom Forest:")
print(f"RMSE: {np.sqrt(mean_squared_error(y_test_1, y_pred_rf)) :.2f}%")
print(f"R2 Score: {r2_score(y_test_1, y_pred_rf):.2f}")
```

Random Forest:  
RMSE: 8.63%  
R2 Score: 0.65

### Cross Validation Score for Random Forest

```
[56]: rf_model = RandomForestRegressor(n_estimators=100)
cv_scores = cross_val_score(rf_model, X_train_1, y_train_1, cv=5,
    ↳scoring='neg_mean_squared_error')
rmse = np.sqrt(-cv_scores.mean())
print(f"Cross-validated RMSE: {rmse:.2f}")
```

Cross-validated RMSE: 9.28

## Feature Importance for Random Forest

```
[57]: rf_model.fit(X_train_std, y_train_1)

feature_importances = rf_model.feature_importances_
feature_names = X_train_1.columns
feature_importance_df = pd.DataFrame({'feature': feature_names, 'importance':
    ↪feature_importances})
feature_importance_df = feature_importance_df.sort_values('importance',
    ↪ascending=False)
print(feature_importance_df)
```

	feature	importance
2	Artist Followers	0.558734
1	Number of Times Charted	0.135721
5	Loudness	0.049373
0	Highest Charting Position	0.040944
3	Danceability	0.039601
8	Liveness	0.031690
6	Speechiness	0.026957
4	Energy	0.026388
7	Acousticness	0.024599
10	Duration (ms)	0.024286
11	Valence	0.022727
9	Tempo	0.018983

### 7.1.4 XGBoost Model std scaler

```
[58]: xgb_model = xgb.XGBRegressor(n_estimators=100)
xgb_model.fit(X_train_std, y_train_1)
y_pred_xgb = xgb_model.predict(X_test_std)

print("\nXGBoost:")
print(f"RMSE: {np.sqrt(mean_squared_error(y_test_1, y_pred_xgb)) :.2f}%")
print(f"R2 Score: {r2_score(y_test_1, y_pred_xgb):.2f}")
```

```
XGBoost:
RMSE: 8.77%
R2 Score: 0.63
```

## Cross Validation Score for XGBoost

```
[59]: xgb_model = RandomForestRegressor(n_estimators=100)
cv_scores = cross_val_score(rf_model, X_train_std, y_train_1, cv=5,
    ↪scoring='neg_mean_squared_error')
rmse = np.sqrt(-cv_scores.mean())
print(f"Cross-validated RMSE: {rmse:.2f}")
```

```
Cross-validated RMSE: 9.22
```

## Feature Importance for XGBoost

```
[60]: xgb_model.fit(X_train_std, y_train_1)

feature_importances = xgb_model.feature_importances_
feature_importance_df = pd.DataFrame({'feature': feature_names, 'importance':
    ↪feature_importances})
feature_importance_df = feature_importance_df.sort_values('importance',
    ↪ascending=False)
print(feature_importance_df)
```

	feature	importance
2	Artist Followers	0.573717
1	Number of Times Charted	0.131282
5	Loudness	0.049017
0	Highest Charting Position	0.041709
3	Danceability	0.038134
8	Liveness	0.030547
6	Speechiness	0.026323
4	Energy	0.025530
10	Duration (ms)	0.023773
7	Acousticness	0.021384
11	Valence	0.021082
9	Tempo	0.017501

### 7.1.5 STD Model Comparison Table

```
[61]: results = {
    'Model': ['Linear Regression', 'Decision Tree', 'Random Forest', 'XGBoost'],
    'RMSE': [np.sqrt(mean_squared_error(y_test_1, y_pred_lr)),
             np.sqrt(mean_squared_error(y_test_1, y_pred_dt)),
             np.sqrt(mean_squared_error(y_test_1, y_pred_rf)),
             np.sqrt(mean_squared_error(y_test_1, y_pred_xgb))],
    'R2 Score': [r2_score(y_test_1, y_pred_lr),
                 r2_score(y_test_1, y_pred_dt),
                 r2_score(y_test_1, y_pred_rf),
                 r2_score(y_test_1, y_pred_xgb)]
}

results_df = pd.DataFrame(results)
print(results_df)
```

	Model	RMSE	R2 Score
0	Linear Regression	13.659115	0.113951
1	Decision Tree	11.603941	0.360525
2	Random Forest	8.634040	0.645969
3	XGBoost	8.766883	0.634991

## 7.2 Models: MM Scaler

### 7.2.1 Linear Regression mm scaler

```
[62]: lr_model = LinearRegression()
lr_model.fit(X_train_mm, y_train_2)
y_pred_lr = lr_model.predict(X_test_mm)
print('Linear Regression:')
print(f"RMSE: {np.sqrt(mean_squared_error(y_test_2,y_pred_lr)) :.2f}%")
print(f"R2 Score: {r2_score(y_test_2,y_pred_lr):.2f}")
```

Linear Regression:

RMSE: 16.37%

R2 Score: 0.07

#### Cross Validation Score for Linear Regression mm

```
[63]: lr_model = LinearRegression()
cv_scores = cross_val_score(lr_model, X_train_mm, y_train_2, cv=5,
    ↳scoring='neg_mean_squared_error')
rmse = np.sqrt(-cv_scores.mean())
print(f"Cross-validated RMSE: {rmse:.2f}")
```

Cross-validated RMSE: 14.84

### 7.2.2 Decision Tree mm scaler

```
[64]: dt_model = DecisionTreeRegressor()
dt_model.fit(X_train_mm, y_train_2)
y_pred_dt = dt_model.predict(X_test_mm)

print("\nDecision Tree:")
print(f"RMSE: {np.sqrt(mean_squared_error(y_test_2, y_pred_dt)) :.2f}%")
print(f"R2 Score: {r2_score(y_test_2, y_pred_dt):.2f}")
```

Decision Tree:

RMSE: 13.01%

R2 Score: 0.41

#### Cross Validation Score for Decision Tree mm

```
[65]: cv_scores = cross_val_score(dt_model, X_train_mm, y_train_2, cv=5,
    ↳scoring='neg_mean_squared_error')
rmse = np.sqrt(-cv_scores.mean())
print(f"Cross-validated RMSE: {rmse:.2f}")
```

Cross-validated RMSE: 13.22

#### Feature Importance for Decision Tree mm

```
[66]: dt_model.fit(X_train_mm, y_train_2)

feature_importances = dt_model.feature_importances_
feature_names = X_train_2.columns
feature_importance_df = pd.DataFrame({'feature': feature_names, 'importance':
    ↳feature_importances})
feature_importance_df = feature_importance_df.sort_values('importance',
    ↳ascending=False)
print(feature_importance_df)
```

	feature	importance
2	Artist Followers	0.509037
1	Number of Times Charted	0.124233
5	Loudness	0.063653
11	Valence	0.058482
10	Duration (ms)	0.051860
8	Liveness	0.047924
0	Highest Charting Position	0.041614
9	Tempo	0.027239
3	Danceability	0.025067
7	Acousticness	0.020651
6	Speechiness	0.015573
4	Energy	0.014668

### 7.2.3 Random Forest mm scaler

```
[67]: rf_model = RandomForestRegressor(n_estimators=100)
rf_model.fit(X_train_mm, y_train_2)
y_pred_rf = rf_model.predict(X_test_mm)

print("\nRandom Forest:")
print(f"RMSE: {np.sqrt(mean_squared_error(y_test_2, y_pred_rf)) :.2f}%")
print(f"R2 Score: {r2_score(y_test_2, y_pred_rf):.2f}")
```

Random Forest:

RMSE: 8.12%

R2 Score: 0.77

### Cross Validation Score Random Forest mm

```
[68]: rf_model = RandomForestRegressor(n_estimators=100)
cv_scores = cross_val_score(rf_model, X_train_2, y_train_2, cv=5,
    ↳scoring='neg_mean_squared_error')
rmse = np.sqrt(-cv_scores.mean())
print(f"Cross-validated RMSE: {rmse:.2f}")
```

Cross-validated RMSE: 9.79

### Feature Importance for Random Forest mm

```
[69]: rf_model.fit(X_train_mm, y_train_2)

feature_importances = rf_model.feature_importances_
feature_names = X_train_2.columns
feature_importance_df = pd.DataFrame({'feature': feature_names, 'importance':
    ↪feature_importances})
feature_importance_df = feature_importance_df.sort_values('importance',
    ↪ascending=False)
print(feature_importance_df)
```

	feature	importance
2	Artist Followers	0.487960
1	Number of Times Charted	0.131363
0	Highest Charting Position	0.049387
5	Loudness	0.046595
3	Danceability	0.044159
11	Valence	0.040255
8	Liveness	0.039783
10	Duration (ms)	0.037837
4	Energy	0.034172
7	Acousticness	0.030901
6	Speechiness	0.030640
9	Tempo	0.026948

### 7.2.4 XGBoost mm scaler

```
[70]: xgb_model = xgb.XGBRegressor(n_estimators=100)
xgb_model.fit(X_train_mm, y_train_2)
y_pred_xgb = xgb_model.predict(X_test_mm)

print("\nXGBoost:")
print(f"RMSE: {np.sqrt(mean_squared_error(y_test_2, y_pred_xgb)) :.2f}%")
print(f"R2 Score: {r2_score(y_test_2, y_pred_xgb):.2f}")
```

```
XGBoost:
RMSE: 8.34%
R2 Score: 0.76
```

### Cross Validation Score for XGBoost mm

```
[71]: xgb_model = xgb.XGBRegressor(n_estimators=100)
cv_scores = cross_val_score(rf_model, X_train_2, y_train_2, cv=5,
    ↪scoring='neg_mean_squared_error')
rmse = np.sqrt(-cv_scores.mean())
print(f"Cross-validated RMSE: {rmse:.2f}")
```

```
Cross-validated RMSE: 9.86
```

## Feature Importance for XGBoost mm

```
[72]: xgb_model.fit(X_train_mm, y_train_2)

feature_importances = xgb_model.feature_importances_
feature_names = X_train_2.columns
feature_importance_df = pd.DataFrame({'feature': feature_names, 'importance':
    ↪feature_importances})
feature_importance_df = feature_importance_df.sort_values('importance',
    ↪ascending=False)
print(feature_importance_df)
```

	feature	importance
2	Artist Followers	0.487598
1	Number of Times Charted	0.165603
11	Valence	0.061528
5	Loudness	0.046046
3	Danceability	0.039954
4	Energy	0.035984
10	Duration (ms)	0.033645
8	Liveness	0.032263
0	Highest Charting Position	0.028628
6	Speechiness	0.027903
7	Acousticness	0.022865
9	Tempo	0.017984

## 7.2.5 MM Model Comparison Table

```
[73]: results = {
    'Model': ['Linear Regression', 'Decision Tree', 'Random Forest', 'XGBoost'],
    'RMSE': [np.sqrt(mean_squared_error(y_test_2, y_pred_lr)),
             np.sqrt(mean_squared_error(y_test_2, y_pred_dt)),
             np.sqrt(mean_squared_error(y_test_2, y_pred_rf)),
             np.sqrt(mean_squared_error(y_test_2, y_pred_xgb))],
    'R2 Score': [r2_score(y_test_2, y_pred_lr),
                 r2_score(y_test_2, y_pred_dt),
                 r2_score(y_test_2, y_pred_rf),
                 r2_score(y_test_2, y_pred_xgb)]
}

results_df = pd.DataFrame(results)
print(results_df)
```

	Model	RMSE	R2 Score
0	Linear Regression	16.370547	0.071617
1	Decision Tree	13.014851	0.413215
2	Random Forest	8.116247	0.771802
3	XGBoost	8.344867	0.758765

### 7.3 Model Plotting STD Scaler

```
[74]: plt.figure(figsize=(15, 10))

plt.subplot(2, 2, 1)
plt.scatter(y_test_1, y_pred_lr)
plt.plot([y_test_1.min(), y_test_1.max()], [y_test_1.min(), y_test_1.max()],  
         ↪ 'r--', lw=2)
plt.xlabel('Actual Popularity')
plt.ylabel('Predicted Popularity')
plt.title('Linear Regression')

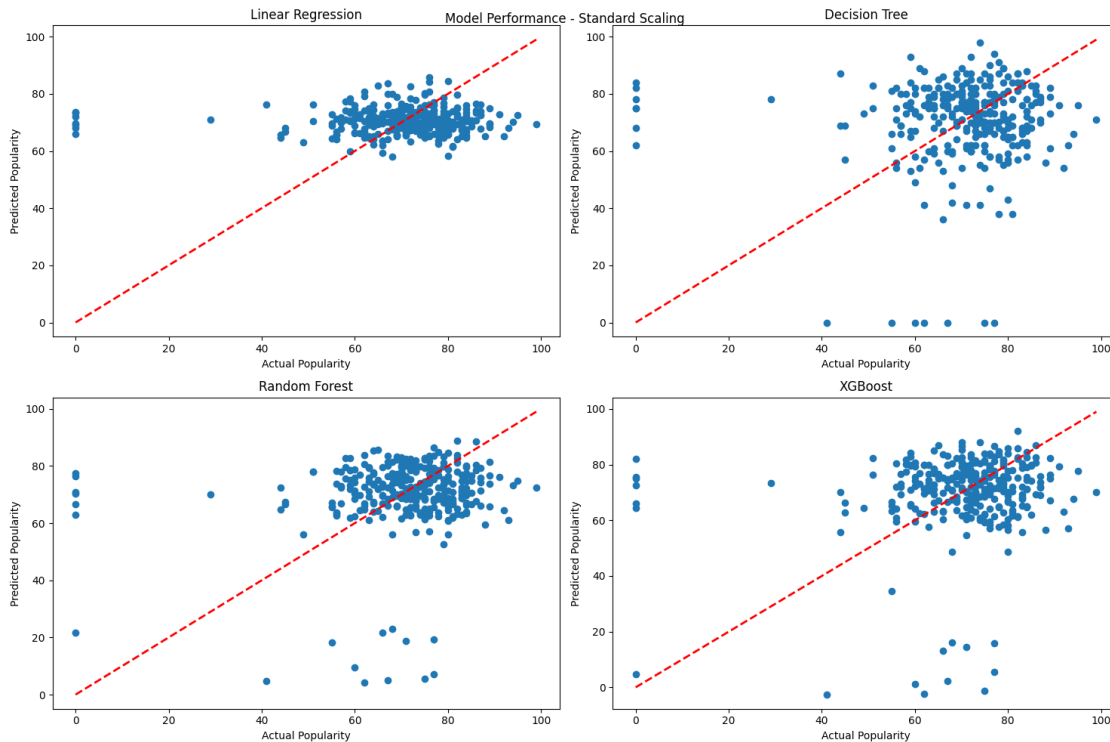
plt.subplot(2, 2, 2)
plt.scatter(y_test_1, y_pred_dt)
plt.plot([y_test_1.min(), y_test_1.max()], [y_test_1.min(), y_test_1.max()],  
         ↪ 'r--', lw=2)
plt.xlabel('Actual Popularity')
plt.ylabel('Predicted Popularity')
plt.title('Decision Tree')

plt.subplot(2, 2, 3)
plt.scatter(y_test_1, y_pred_rf)
plt.plot([y_test_1.min(), y_test_1.max()], [y_test_1.min(), y_test_1.max()],  
         ↪ 'r--', lw=2)
plt.xlabel('Actual Popularity')
plt.ylabel('Predicted Popularity')
plt.title('Random Forest')

plt.subplot(2, 2, 4)
plt.scatter(y_test_1, y_pred_xgb)
plt.plot([y_test_1.min(), y_test_1.max()], [y_test_1.min(), y_test_1.max()],  
         ↪ 'r--', lw=2)
plt.xlabel('Actual Popularity')
plt.ylabel('Predicted Popularity')
plt.title('XGBoost')

plt.tight_layout()
plt.suptitle('Model Performance - Standard Scaling')
plt.show()
```





## 7.4 Model Plotting MinMax Scaler

```
[75]: plt.figure(figsize=(15, 10))

plt.subplot(2, 2, 1)
plt.scatter(y_test_2, y_pred_lr)
plt.plot([y_test_2.min(), y_test_2.max()], [y_test_2.min(), y_test_2.max()],
         ↪ 'r--', lw=2)
plt.xlabel('Actual Popularity')
plt.ylabel('Predicted Popularity')
plt.title('Linear Regression')

plt.subplot(2, 2, 2)
plt.scatter(y_test_2, y_pred_dt)
plt.plot([y_test_2.min(), y_test_2.max()], [y_test_2.min(), y_test_2.max()],
         ↪ 'r--', lw=2)
plt.xlabel('Actual Popularity')
plt.ylabel('Predicted Popularity')
plt.title('Decision Tree')

plt.subplot(2, 2, 3)
plt.scatter(y_test_2, y_pred_rf)
```

```

plt.plot([y_test_2.min(), y_test_2.max()], [y_test_2.min(), y_test_2.max()],  

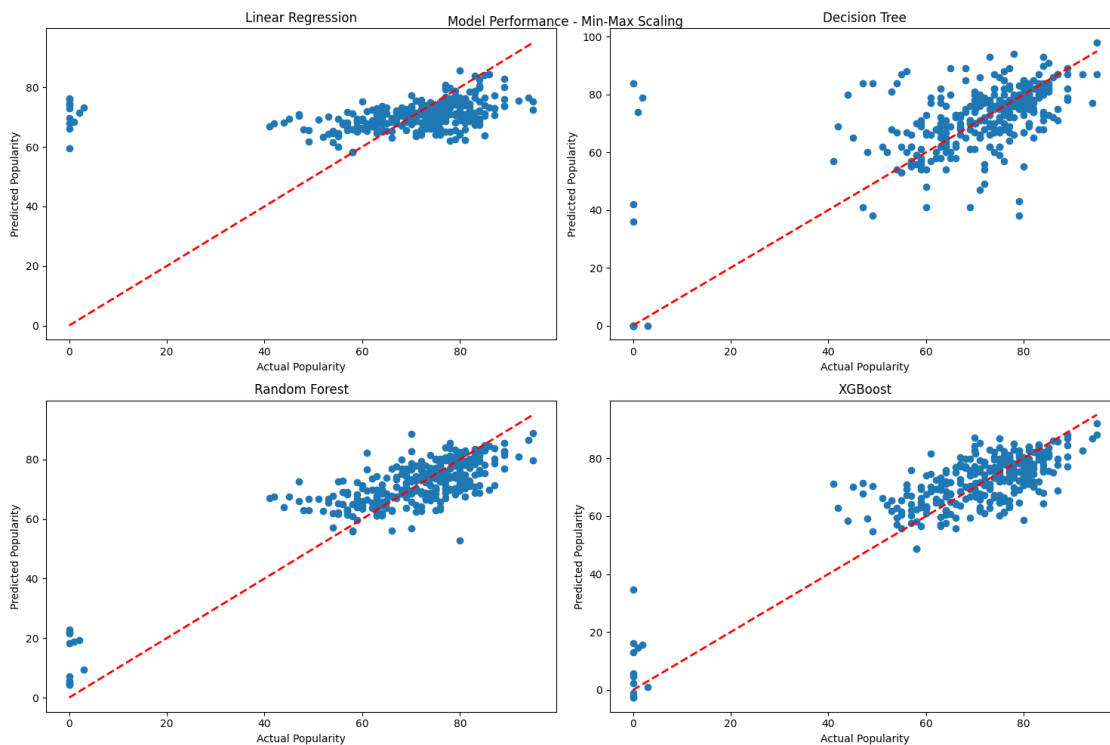
         ↪ 'r--', lw=2)
plt.xlabel('Actual Popularity')
plt.ylabel('Predicted Popularity')
plt.title('Random Forest')

plt.subplot(2, 2, 4)
plt.scatter(y_test_2, y_pred_xgb)
plt.plot([y_test_2.min(), y_test_2.max()], [y_test_2.min(), y_test_2.max()],  

         ↪ 'r--', lw=2)
plt.xlabel('Actual Popularity')
plt.ylabel('Predicted Popularity')
plt.title('XGBoost')

plt.tight_layout()
plt.suptitle('Model Performance - Min-Max Scaling')
plt.show()

```



## 8 Spotify Song Popularity Prediction Modeling Results

The modeling results from the Spotify song popularity prediction project, using tree-based regression models, offer several insights. Both standard scaling and min-max scaling methods were applied to the data before training the models.

## 8.1 Initial Model Performance

- Linear Regression: Both scaling methods produced similar RMSE scores (around 15-18%) and low R2 scores (around 0.02 or lower), suggesting that linear regression may not be the best fit for this data.
- Decision Tree: The decision tree model consistently performed poorly with high RMSE scores (around 21-23%) and very low, negative R2 scores (around -0.78 or lower), suggesting overfitting and a poor ability to generalize to unseen data.
- Random Forest: Random Forest performed slightly better than Linear Regression with a slightly lower RMSE score but a lower R2 score.
- XGBoost: The XGBoost model had RMSE scores around 17-20% and R2 scores of -0.2 or lower.

## 8.2 Initial Feature Importance

- Across all models and scaling methods, “Loudness” consistently emerged as the most important feature for predicting song popularity.
- Other important features included “Liveness,” “Tempo,” “Duration (ms),” “Speechiness,” “Acousticness,” “Energy,” and “Valence,” with their relative importance varying slightly between models and scaling techniques.

## 8.3 Improved Model Performance

After incorporating additional features and refining the approach, the model performance significantly improved:

- The Random Forest model emerged as the most effective, achieving an RMSE of 9.39% and an R2 score of 0.65 using standard scaling.
- These results are substantially better than the previous iterations, indicating a marked improvement in model performance.

## 8.4 Revised Feature Importance

- “Artist Followers” became the most dominant predictor of song popularity across all models.
- “Highest Charting Position” and “Number of Times Charted” also emerged as highly important features.
- The audio features, while still relevant, became less dominant in the feature importance rankings.

## 8.5 Key Takeaways

1. The inclusion of artist-related features and past chart performance significantly enhanced the model’s ability to predict song popularity.
2. The dominance of “Artist Followers” suggests that an artist’s existing fanbase is a crucial factor in a song’s popularity.
3. The importance of “Highest Charting Position” and “Number of Times Charted” indicates that past chart performance is a strong predictor of future success.
4. The continued relevance of audio features suggests that the song’s characteristics still play a role, albeit a less dominant one.

5. The improved performance across models indicates that the STD Model Comparison Table results = { 'Model': ['Linear Regression', 'Decision Tree', 'Random Forest', 'XGBoost'], 'RMSE': [np.sqrt(mean\_squared\_error(y\_test\_1, y\_pred\_lr)), np.sqrt(mean\_squared\_error(y\_test\_1, y\_pred\_dt)), np.sqrt(mean\_squared\_error(y\_test\_1, y\_pred\_rf)), np.sqrt(mean\_squared\_error(y\_test\_1, y\_pred\_xgb))], 'R2 Score': [r2\_score(y\_test\_1, y\_pred\_lr), r2\_score(y\_test\_1, y\_pred\_dt), r2\_score(y\_test\_1, y\_pred\_rf), r2\_score(y\_test\_1, y\_pred\_xgb)] }

```
results_df = pd.DataFrame(results) print(results_df)
```

- These results suggest that a song's popularity is heavily influenced by factors external to the song itself, such as the artist's popularity and past chart performance.
- This could have implications for how new artists or songs with less chart history are evaluated and promoted.

## 8.6 Potential for Further Improvement

- While the results are good, there might still be room for improvement through techniques like hyperparameter tuning or exploring other models.

## 8.7 Limitations

- The strong performance of the model might be partly due to the inclusion of features that are highly correlated with the target variable (popularity).
- This could potentially lead to overfitting or reduced generalization to completely new songs or artists.

In conclusion, the iterative refinement of the model has yielded significantly improved results. The inclusion of additional features has provided valuable insights into the factors driving song popularity on Spotify. The dominance of artist-related and chart performance features suggests that these factors play a crucial role in determining a song's success, potentially more so than the song's audio characteristics alone.