Spotify.Description.for.Students

November 5, 2024

1 Project 4: Music Popularity Prediction

This project will take data features collected for songs that have been on the Top 200 Weekly (Global) charts of Spotify in 2020 & 2021. The popularity of the song will be predicted using a tree-based regression model trained on these features.

The goals for the project are:

- Minimize the cross-validated ***root mean squared error (RMSE)*** when predicting the popularity of a new song.
- Determine the importance of the features in driving the regression result. The project will be done using tree-based regression techniques as covered in class. The parameters of the trees should be carefully selected to avoid over-fitting.

There are three main challenges for this project:

1. Determining the outcome (i.e. target). There is a "popularity" column. But other columns may or may not be more appropriate indicators of popularity.
2. Choosing appropriate predictors (i.e. features). When building a machine learning model, we want to make sure that we consider how the model will be ultimately used. For this project, we are predicting the popularity of a new song. Therefore, we should only include the predictors we would have for a new song.
3. Data cleaning and feature engineering. Some creative cleaning and/or feature engineering may be needed to extract useful information for prediction.

Once again, be sure to go through the whole data science process and document as such in your Jupyter notebook.

The data is available AWS at https://ddc-datascience.s3.amazonaws.com/Projects/Project.4-Spotify/Data/Spotify.csv .

2 Imports

[811]:
```
import sys
print(sys.executable)
```

```
/usr/local/bin/python
```

[915]:
```
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
```

[813]:
```
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
```

[814]:
```
%%capture
url = "https://ddc-datascience.s3.amazonaws.com/Projects/Project.4-Spotify/Data/
 ↪Spotify.csv"
!curl -s -I {url}
```

3 Data Exploration

[815]:
```
df_1 = pd.read_csv(url).copy()
```

3.1 Head

[816]:
```
df_1.head()
```

[816]:

	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  27NovPIUIRrOZoCHxABJwK

                                  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

[817]:
```
df_1.shape
```

[817]:
```
(1556, 23)
```

3.4 columns

[818]:
```
df_1.columns
```

[818]:
```
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

[819]:
```
df_1.dtypes
```

[819]:
```
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

[820]:
```
df_1.describe()
```

[820]:

	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

[821]:
```
df_1.isnull().sum()
```

[821]:
```
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

[822]:
```
df_1.isna().sum()
```

[822]:
```
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

[823]:
```
df_1.count('rows').unique().sum()
```

[823]:
```
np.int64(1556)
```

[824]:
```
df_1.count('columns')
```

[824]:
```
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

[825]:
```
df_1.sort_values(by = ['Popularity'], ascending = False).head(10)
```

[825]:

	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	4ZtFanR9U6ndgddUvNcjcG
3	Ed Sheeran	83293380	6PQ88X9TkUIAUIZJHW2upE
5	Lil Nas X	5473565	67BtfxlNbhBmCDR2L2l8qd
4	Lil Nas X	5473565	27NovPIUIRrOZoCHxABJwK
8	Bad Bunny	36142273	2JPLbjOn0wPCngEot2STUS
14	BTS	37106176	2bgTY4UwhfBYhGT4HUYStN
7	Rauw Alejandro	6080597	4fSIb4hdOQ151TILNsSEaF
9	Måneskin	3377762	4pt5fDVTg5GhEvEtlz9dKk
11	J Balvin, Maria Becerra	29051363	6hf0RpxTbOprT5nnwzkk8e

	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

[826]:
```
df_cleaning = df_1.copy()
df_cleaning
```

[826]:
```
      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	3Wrjm47oTz2sjIgck11l5e	
1	The Kid LAROI	2230022	5HCyWlXZPP0y6Gqq8TgA20	
2	Olivia Rodrigo	6266514	4ZtFanR9U6ndgddUvNcjcG	
3	Ed Sheeran	83293380	6PQ88X9TkUIAUIZJHW2upE	
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	3i9UVldZOE0aD0JnyfAZZ0	

	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

[827]:
```
df_cleaning.drop('Index', axis = 1, inplace = True)
#
```

[828]:
```
df_cleaning.transpose()
```

[828]:

	0	1
Highest Charting Position	1	2
Number of Times Charted	8	3
Week of Highest Charting	2021-07-23--2021-07-30	2021-07-23--2021-07-30
Song Name	Beggin'	STAY (with Justin Bieber)
Streams	48,633,449	47,248,719
Artist	Måneskin	The Kid LAROI
Artist Followers	3377762	2230022
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	

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	27NovPIUIRrOZoCHxABJwK
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                                            67BtfxlNbhBmCDR2L2l8qd
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                                            748mdHapucXQri7IAO8yFK
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	4fSIb4hdOQ151TILNsSEaF
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	2JPLbjOn0wPCngEot2STUS
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	…	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                                   3i9UVldZOE0aD0JnyfAZZ0
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

[829]:
```
# List of columns to convert
columns_to_convert = ['Artist Followers', 'Streams','Popularity',␣
 ↪'Danceability', 'Energy', 'Loudness',
                      'Speechiness', 'Acousticness', 'Liveness', 'Tempo',␣
 ↪'Duration (ms)', 'Valence']

# Convert columns to numeric
for column in columns_to_convert:
    df_1[column] = pd.to_numeric(df_1[column], errors='coerce')
```

[830]:
```
df_1.dtypes
```

[830]:
```
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

[831]:
```
df_1 = df_1.drop("Index", axis = 1)
```

[832]:
```
df_1
```

[832]:
```
      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  2PWjKmjyTZeDpmOUa3a5da
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]
```

[833]:
```
df_clean_2 = df_1.copy()
```

5.1 Identify Object Columns & Drop them

[834]:
```
object_columns = df_clean_2.select_dtypes(include=['object']).columns
df_clean_2 = df_clean_2.drop(columns=object_columns)
```

[835]:
```
df_clean_2.isnull().sum()
```

[835]:
```
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
```

[836]:
```
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)

[837]:
```
df_clean_2.drop('Streams', axis = 1, inplace = True)
```

[838]:
```
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

[839]:
```
df_clean_2.isna().sum()
```

[839]:
```
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
```

[840]:
```
null_columns = df_clean_2.columns[df_clean_2.isnull().any()].tolist()
print("Columns with null values:")
null_columns
```

```
Columns with null values:
```

[840]:
```
['Artist Followers',
 'Popularity',
 'Danceability',
 'Energy',
 'Loudness',
 'Speechiness',
 'Acousticness',
 'Liveness',
 'Tempo',
 'Duration (ms)',
 'Valence']
```

[841]:
```
for col in null_columns:
    #Calculate the mean, exluding 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)
```

[842]:
```
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
```

[843]:
```
df_clean_2.dtypes
```

[843]:
```
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"

[844]:
```
df_clean_2.drop('Highest Charting Position', axis = 1, inplace = True)
```

[845]:
```
df_clean_2.drop('Number of Times Charted', axis = 1, inplace = True)
```

[846]:
```
df_clean_2.drop('Artist Followers', axis = 1, inplace = True)
```

[847]:
```
df_scaling = df_clean_2.copy()
```

6 Data Scaling

6.1 Data Scaling (standard scaler)

6.1.1 Setup standard scaled training and testing data

[848]:
```
df_3_std = df_scaling.copy()
```

[849]:
```
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)
```

[850]:
```
scaler = StandardScaler()
X_train_std = scaler.fit_transform(X_train_1)
X_test_std = scaler.transform(X_test_1)
```

[851]:
```
print("Before scaling:")
print(X_train_1.describe())

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

```
Before scaling:
       Danceability       Energy    Loudness  Speechiness  Acousticness  \
count   1244.000000  1244.000000  1244.00000  1244.000000   1244.000000
mean       0.689711     0.636668    -6.29766     0.124709      0.244569
std        0.142779     0.160244     2.41891     0.110808      0.248308
min        0.150000     0.054000   -25.16600     0.023200      0.000025
25%        0.599000     0.536000    -7.46525     0.045575      0.047475
50%        0.703000     0.641500    -5.96150     0.078800      0.158500
75%        0.795250     0.757250    -4.67100     0.169250      0.371000
max        0.980000     0.960000     1.50900     0.884000      0.979000

          Liveness        Tempo  Duration (ms)      Valence
count  1244.000000  1244.000000    1244.000000  1244.000000
mean      0.183574   122.823524  197440.422818     0.515912
std       0.142768    29.515437   44553.839282     0.226466
min       0.027300    62.948000   30133.000000     0.032000
25%       0.097975    97.962500  170317.000000     0.344000
50%       0.127000   122.054500  193854.000000     0.512500
75%       0.224000   143.553500  218117.000000     0.692500
max       0.962000   205.272000  484147.000000     0.977000

After scaling:
                  0             1             2             3             4  \
count  1.244000e+03  1.244000e+03  1.244000e+03  1.244000e+03  1.244000e+03
mean   3.926834e-16  4.398054e-16  1.770645e-16  9.138813e-17 -1.199469e-16
std    1.000402e+00  1.000402e+00  1.000402e+00  1.000402e+00  1.000402e+00
min   -3.781568e+00 -3.637588e+00 -7.803485e+00 -9.164525e-01 -9.852352e-01
25%   -6.355829e-01 -6.284708e-01 -4.828867e-01 -7.144449e-01 -7.940670e-01
50%    9.310865e-02  3.016381e-02  1.390276e-01 -4.144805e-01 -3.467611e-01
75%    7.394720e-01  7.527890e-01  6.727470e-01  4.021267e-01  5.093751e-01
max    2.033950e+00  2.018554e+00  3.228644e+00  6.855084e+00  2.958932e+00

                  5             6             7             8
count  1.244000e+03  1.244000e+03  1.244000e+03  1.244000e+03
mean   1.713527e-17 -2.584571e-16  3.555570e-16 -4.783598e-17
```

```
std    1.000402e+00  1.000402e+00  1.000402e+00  1.000402e+00
min   -1.095041e+00 -2.029433e+00 -3.756684e+00 -2.137654e+00
25%   -5.998075e-01 -8.426446e-01 -6.090234e-01 -7.594109e-01
50%   -3.964235e-01 -2.606547e-02 -8.052875e-02 -1.507131e-02
75%    2.832749e-01  7.026260e-01  4.642673e-01  7.800689e-01
max    5.454589e+00  2.794525e+00  6.437647e+00  2.036832e+00
```

[852]:
```
print("Mean:", X_train_std.mean(axis=0))
print("Std:", X_train_std.std(axis=0))
```

```
Mean: [ 3.92683385e-16  4.39805391e-16  1.77064508e-16  9.13881332e-17
 -1.19946925e-16  1.71352750e-17 -2.58457064e-16  3.55556956e-16
 -4.78359760e-17]
Std: [1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

6.2 Data Scaling Continued (min-max scaler)

[853]:
```
df_3_mm = df_scaling.copy()
```

[854]:
```
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

[855]:
```
scaler = MinMaxScaler()
X_train_mm = scaler.fit_transform(X_train_2)
X_test_mm = scaler.transform(X_test_2)
```

[856]:
```
print("Before scaling:")
print(X_train_2.describe())

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

```
Before scaling:
       Danceability       Energy     Loudness  Speechiness  Acousticness  \
count   1244.000000  1244.000000  1244.000000  1244.000000   1244.000000
mean       0.692152     0.632379    -6.349187     0.122211      0.249215
std        0.143242     0.159891     2.458677     0.108784      0.248820
min        0.150000     0.054000   -25.166000     0.023200      0.000025
25%        0.601750     0.533750    -7.481500     0.045975      0.049200
50%        0.709500     0.639000    -6.013500     0.077050      0.163500
75%        0.800250     0.747000    -4.748750     0.163000      0.388250
max        0.980000     0.970000    -0.515000     0.884000      0.991000
```

```
          Liveness        Tempo  Duration (ms)      Valence
count  1244.000000  1244.000000    1244.000000  1244.000000
mean      0.179323   122.332108  198791.797054     0.513455
std       0.141902    29.559311   47731.842587     0.227259
min       0.019700    62.948000   30133.000000     0.036000
25%       0.096575    97.012500  169855.750000     0.344000
50%       0.123500   122.012000  193834.000000     0.512500
75%       0.209250   142.626750  219418.250000     0.688250
max       0.923000   205.272000  588139.000000     0.979000

After scaling:
                 0            1            2            3            4  \
count  1244.000000  1244.000000  1244.000000  1244.000000  1244.000000
mean      0.653195     0.631418     0.763329     0.115022     0.251459
std       0.172581     0.174554     0.099739     0.126375     0.251086
min       0.000000     0.000000     0.000000     0.000000     0.000000
25%       0.544277     0.523745     0.717395     0.026458     0.049622
50%       0.674096     0.638646     0.776946     0.062558     0.164963
75%       0.783434     0.756550     0.828252     0.162407     0.391760
max       1.000000     1.000000     1.000000     1.000000     1.000000

                 5            6            7            8
count  1244.000000  1244.000000  1244.000000  1244.000000
mean      0.176711     0.417246     0.302253     0.506315
std       0.157093     0.207690     0.085540     0.240996
min       0.000000     0.000000     0.000000     0.000000
25%       0.085105     0.239345     0.250397     0.326617
50%       0.114912     0.414997     0.293368     0.505302
75%       0.209842     0.559841     0.339217     0.691676
max       1.000000     1.000000     1.000000     1.000000
```

[857]:
```
print("Mean:", X_train_mm.mean(axis=0))
print("Std:", X_train_mm.std(axis=0))
```

```
Mean: [0.65319507 0.63141767 0.76332858 0.11502176 0.25145901 0.17671113
 0.41724592 0.30225266 0.50631471]
Std: [0.17251138 0.17448374 0.09969933 0.12632457 0.25098487 0.15703007
 0.20760679 0.08550563 0.24089877]
```

7 Model Selection and Training

7.1 Models: STD Scaler

7.1.1 Linear Regression std scaler

[858]:
```
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: 17.81%
R2 Score: -0.05
```

Cross Validation Score for Linear Regression

[859]:
```
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: 15.01
```

7.1.2 Decision Tree Model std scaler

[860]:
```
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: 23.22%
R2 Score: -0.78
```

Cross Validation Score for Decision Tree

[861]:
```
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: 22.36
```

Feature Importance for Decision Tree

[862]:
```
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       Loudness    0.148751
1         Energy    0.131105
5       Liveness    0.125242
3    Speechiness    0.115714
6          Tempo    0.110568
7  Duration (ms)    0.104789
4   Acousticness    0.101971
0   Danceability    0.083141
8        Valence    0.078718
```

7.1.3 Random Forest Model std scaler

[863]:
```
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: 18.75%
R2 Score: -0.16
```

Cross Validation Score for Random Forest

[864]:
```
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: 15.36
```

Feature Importance for Random Forest

[865]:
```
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       Loudness    0.156603
4   Acousticness    0.114896
8        Valence    0.111829
6          Tempo    0.110421
5       Liveness    0.109712
3    Speechiness    0.107076
1         Energy    0.104520
7  Duration (ms)    0.097773
0   Danceability    0.087169
```

7.1.4 XGBoost Model std scaler

[866]:
```
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: 19.70%
R2 Score: -0.28
```

Cross Validation Score for XGBoost

[867]:
```
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: 15.38
```

Feature Importance for XGBoost

[868]:
```
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       Loudness    0.160955
8        Valence    0.111458
5       Liveness    0.111040
4   Acousticness    0.109581
1         Energy    0.106445
3    Speechiness    0.106136
7  Duration (ms)    0.102611
6          Tempo    0.102379
0   Danceability    0.089394
```

7.1.5 STD Model Comparison Table

[869]:
```
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  17.812750 -0.049766
1      Decision Tree  23.222462 -0.784214
2      Random Forest  18.753369 -0.163560
3            XGBoost  19.698680 -0.283821
```

7.2 Models: MM Scaler

7.2.1 Linear Regression mm scaler

[870]:
```
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: 15.72%
R2 Score: 0.02
```

Cross Validation Score for Linear Regression mm

[871]:
```
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: 15.58
```

7.2.2 Decision Tree mm scaler

[872]:
```
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: 21.59%
R2 Score: -0.84
```

Cross Validation Score for Decision Tree mm

[873]:
```
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: 23.14
```

Feature Importance for Decision Tree mm

[874]:
```
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       Loudness    0.186728
5       Liveness    0.148189
6          Tempo    0.141205
7  Duration (ms)    0.134782
3    Speechiness    0.107141
4   Acousticness    0.088411
1         Energy    0.077017
0   Danceability    0.061259
8        Valence    0.055266
```

7.2.3 Random Forest mm scaler

[875]:
```
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: 15.96%
R2 Score: -0.01
```

Cross Validation Score Random Forest mm

[876]:
```
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: 16.05
```

Feature Importance for Random Forest mm

[877]:
```
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       Loudness    0.153460
5       Liveness    0.130090
7  Duration (ms)    0.111126
8        Valence    0.108546
1         Energy    0.107864
3    Speechiness    0.100912
6          Tempo    0.100526
0   Danceability    0.095174
4   Acousticness    0.092302
```

7.2.4 XGBoost mm scaler

[878]:
```
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: 17.46%
R2 Score: -0.20
```

Cross Validation Score for XGBoost mm

[879]:
```
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: 16.11
```

Feature Importance for XGBoost mm

[880]:
```
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       Loudness    0.152290
3    Speechiness    0.128658
5       Liveness    0.121053
7  Duration (ms)    0.121003
8        Valence    0.115735
6          Tempo    0.104622
1         Energy    0.098737
4   Acousticness    0.094898
0   Danceability    0.063004
```

7.2.5 MM Model Comparison Table

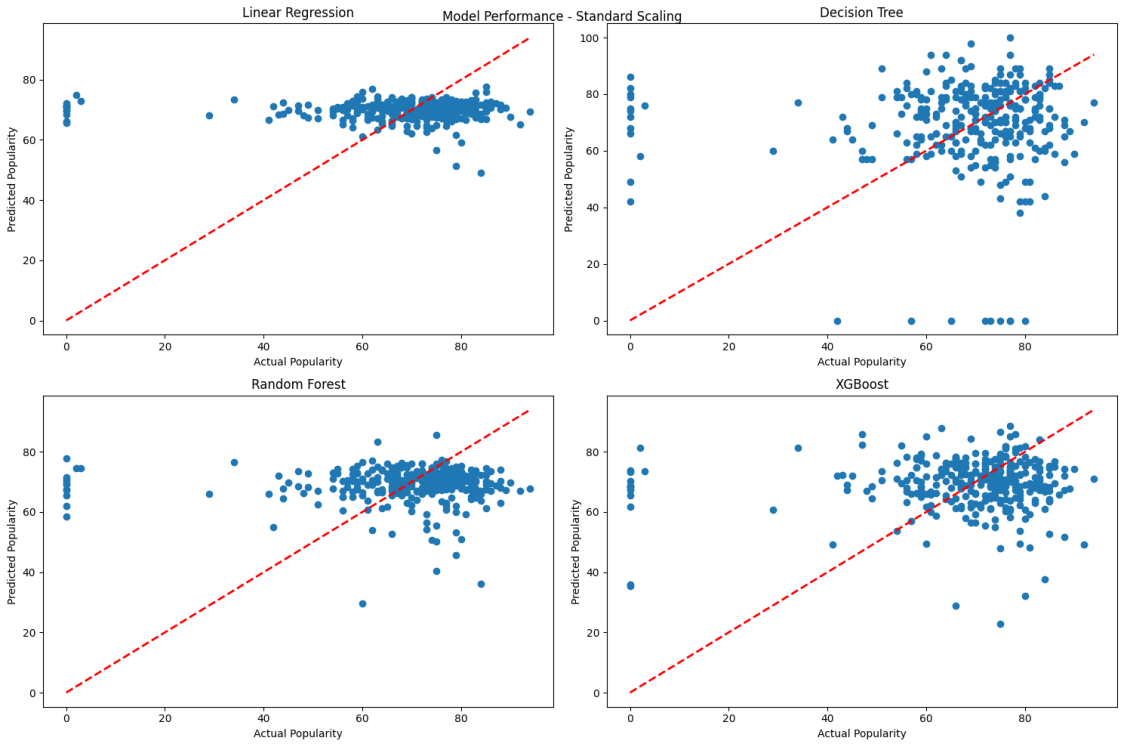
[881]:
```
results = {
    'Model': ['Linear Regression', 'Decision Tree', 'Random Forest', 'XGBoost'],
    'RMSE': [np.sqrt(mean_squared_error(y_test_2, y_pred_lr)),
             np.sqrt(mean_squared_error(y_test_2, y_pred_dt)),
             np.sqrt(mean_squared_error(y_test_2, y_pred_rf)),
             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  15.716834  0.023124
1      Decision Tree  21.590745 -0.843506
2      Random Forest  15.958920 -0.007202
3            XGBoost  17.455136 -0.204914
```

7.3 Model Plotting STD Scaler

[882]:
```
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)
```

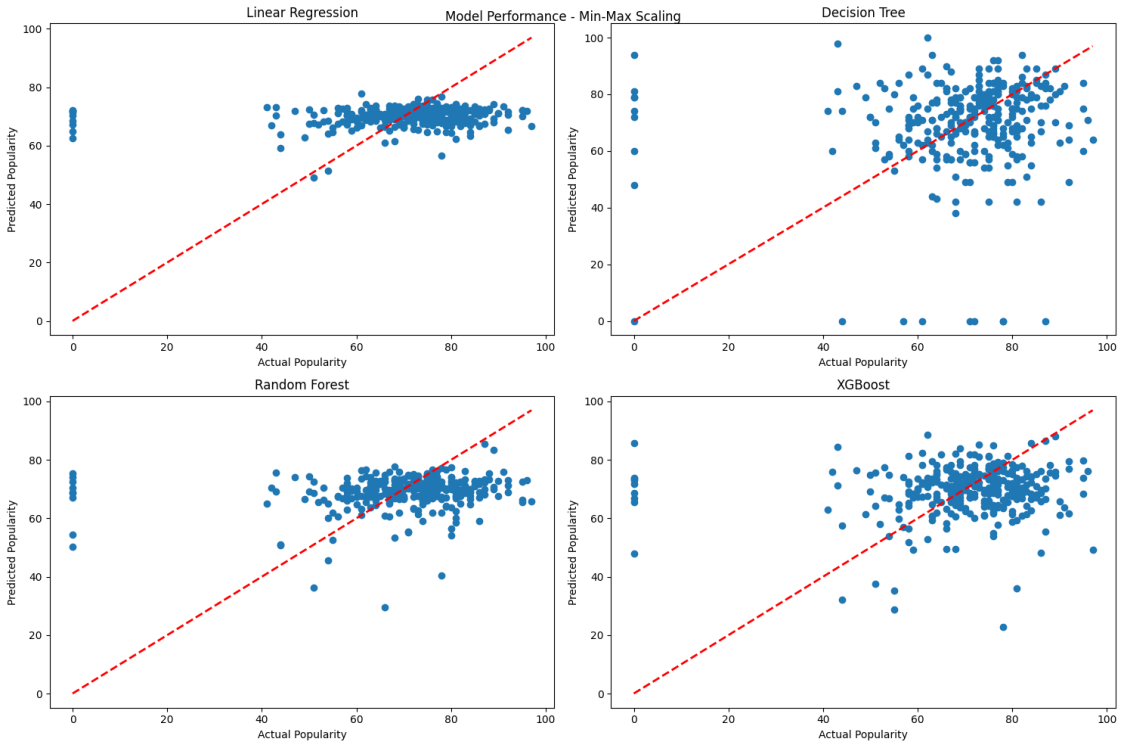
```
plt.xlabel('Actual Popularity')
plt.ylabel('Predicted Popularity')
plt.title('Linear Regression')

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

[918]:
```
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()
```



7.5 Highest Correlated Features by Model and Scaling type

[919]:
```
# Standard Scaling
dt_importance_std = dt_model.feature_importances_
rf_importance_std = rf_model.feature_importances_
xgb_importance_std = xgb_model.feature_importances_
```

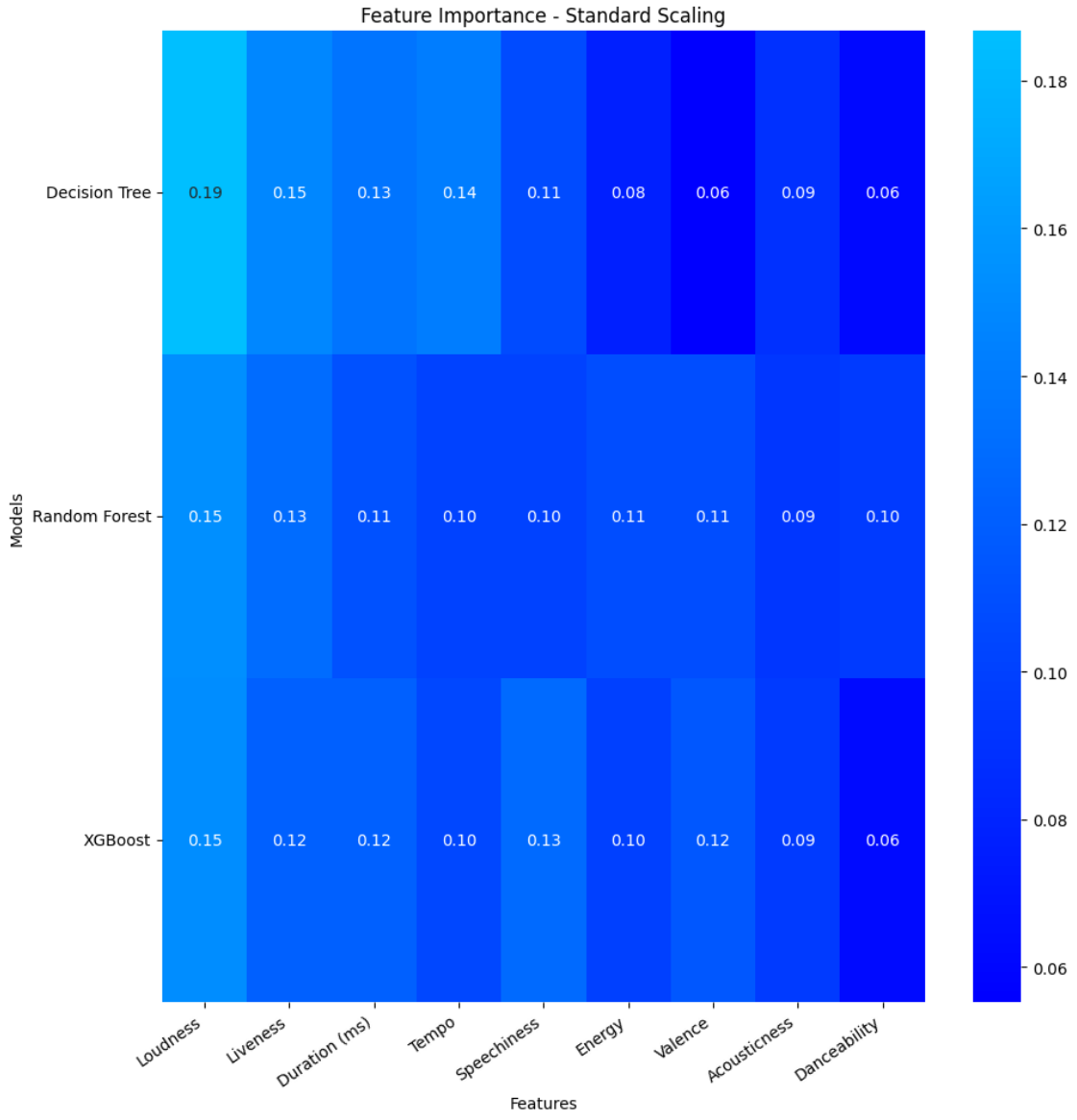
[920]:
```
# Min-Max Scaling
dt_importance_mm = dt_model.feature_importances_
rf_importance_mm = rf_model.feature_importances_
xgb_importance_mm = xgb_model.feature_importances_
```

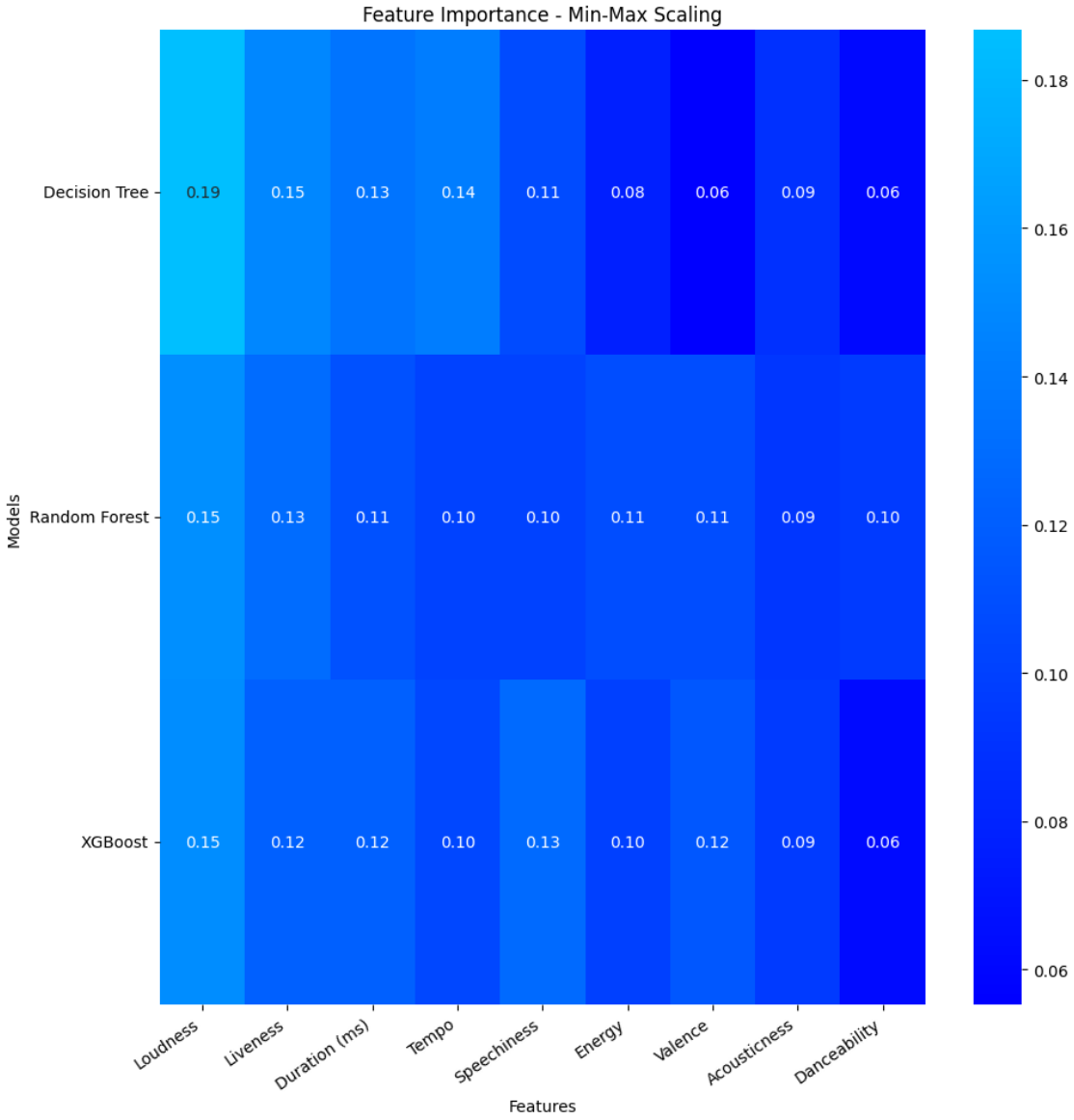
[921]:
```
feature_names = X_train_1.columns
```

[932]:
```
def plot_feature_importance(importances, feature_names, model_names, title):
    plt.figure(figsize=(10, 10), )

    # Create a DataFrame with feature importances
    df = pd.DataFrame(importances, index=model_names, columns=feature_names)

    # Sort features by average importance across all models
    avg_importance = df.mean()
    sorted_features = avg_importance.sort_values(ascending=False).index

    # Custom Color Map
    colors = ["#0000FF", "#00BFFF"]  # Blue to Cerulean
    n_bins = 100
    cmap = mcolors.LinearSegmentedColormap.from_list("custom", colors, N=n_bins)


    # Create heatmap
    sns.heatmap(df[sorted_features], annot=True, cmap=cmap, fmt='.2f')

    plt.title(title)
    plt.xlabel('Features')
    plt.ylabel('Models')
    plt.xticks(rotation=35, ha='right')
    plt.yticks(rotation=0)
    plt.tight_layout()
    plt.show()
```

[933]:
```
# Standard Scaling
importances_std = [dt_importance_std, rf_importance_std, xgb_importance_std]
model_names = ['Decision Tree', 'Random Forest', 'XGBoost']
plot_feature_importance(importances_std, feature_names, model_names, 'Feature␣
↪Importance - Standard Scaling')
```

```
# Min-Max Scaling
importances_mm = [dt_importance_mm, rf_importance_mm, xgb_importance_mm]
plot_feature_importance(importances_mm, feature_names, model_names, 'Feature␣
 ↪Importance - Min-Max Scaling')
```




Feature Importance - Min-Max Scaling
Decision Tree
Random Forest
XGBoost
Models
0.19 0.15 0.13 0.14 0.11 0.08 0.06 0.09 0.06
0.15 0.13 0.11 0.10 0.10 0.11 0.11 0.09 0.10
0.15 0.12 0.12 0.10 0.13 0.10 0.12 0.09 0.06
Loudness
Liveness
Duration (ms)
Tempo
Speechiness
Energy
Valence
Acousticness
Danceability
Features
0.18
0.16
0.14
0.12
0.10
0.08
0.06