Decoding Success at Eurovision: A Machine Learning Analysis of Song Topics, Performance Attributes, and Outcomes (2016–2025)

Data Collection and Inspection-result dataframe

Data Collection

```
import kagglehub

# Download latest version of dataset of Eurovision result from Kaggie
path = kagglehub.dataset_download("rhyspeploe/eurovision-2016-25")

print("Path to dataset files:", path)
```

```
Downloading from https://www.kaggle.com/api/v1/datasets/download/rhyspeploe
100%|██████████| 14.6k/14.6k [00:00<00:00, 18.5MB/s]Extracting files...
Path to dataset files: /root/.cache/kagglehub/datasets/rhyspeploe/eurovisic
```

```
# Import dataset to Google Colab Environment, please uploadda here the file name
from google.colab import files
uploaded = files.upload()
```

Choose Files eurovision_2016-25.csv

- **eurovision_2016-25.csv**(text/csv) - 44257 bytes, last modified: n/a - 100% done

```
Saving eurovision_2016-25.csv to eurovision_2016-25.csv
```

```
# Read in the result file
import pandas as pd
result = pd.read_csv(
    "eurovision_2016-25.csv",
    encoding="ISO-8859-1"
)
```

Data Inspection

```
# Inspect the head
result.head()
```

	Year	Country	Song	Artist	Final_Place	Final_Points	Top 5	Top 10	I
0	2025	Albania	Zjerm	Shkodra Elektronike	8.0	218.0	0.0	1.0	
1	2025	Armenia	Survivor	Parg	20.0	72.0	0.0	0.0	
2	2025	Australia	Milkshake Man	Go-Jo	NaN	NaN	NaN	NaN	
3	2025	Austria	Wasted Love	JJ	1.0	436.0	1.0	1.0	
4	2025	Azerbaijan	Run with U	Mamagama	NaN	NaN	NaN	NaN	

5 rows × 33 columns

```
# Check basic information of the result dataset
result.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 358 entries, 0 to 357
Data columns (total 33 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Year                    358 non-null    int64
 1   Country                 358 non-null    object
 2   Song                    358 non-null    object
 3   Artist                  358 non-null    object
 4   Final_Place             232 non-null    float64
 5   Final_Points            232 non-null    float64
 6   Top 5                   304 non-null    float64
 7   Top 10                  309 non-null    float64
 8   Running_Order_Final     233 non-null    float64
 9   Grand_Final_Ind         358 non-null    int64
 10  Big6_Ind                358 non-null    int64
 11  Semi_Final_Num          305 non-null    float64
 12  Semi_Place              305 non-null    float64
 13  Semi_Points             305 non-null    float64
 14  Running_Order_Semi      305 non-null    float64
 15  National_Final          358 non-null    int64
 16  Solo_Artist             358 non-null    int64
 17  Sex                     358 non-null    object
 18  Returning_Artist_Ind    358 non-null    int64
 19  Number of Members       358 non-null    int64
 20  Language1               358 non-null    object
 21  Language2               68 non-null     object
 22  Language3               7 non-null      object
 23  Language4               2 non-null      object
 24  Multiple_Language       358 non-null    int64
 25  National_Language_Used  358 non-null    bool
 26  EU                      358 non-null    int64
 27  NATO                    358 non-null    int64
 28  Country_Group           358 non-null    object
 29  MyESB_Community         358 non-null    int64
 30  MyESB_Personal          358 non-null    int64
 31  OGAE_Points             358 non-null    int64
 32  Qualification_Record    312 non-null    float64
dtypes: bool(1), float64(10), int64(13), object(9)
memory usage: 90.0+ KB
```

```
# Inspect for missing value
result.isna().sum()
```

	0
Year	0
Country	0

Country	0
Song	0
Artist	0
Final_Place	126
Final_Points	126
Top 5	54
Top 10	49
Running_Order_Final	125
Grand_Final_Ind	0
Big6_Ind	0
Semi_Final_Num	53
Semi_Place	53
Semi_Points	53
Running_Order_Semi	53
National_Final	0
Solo_Artist	0
Sex	0
Returning_Artist_Ind	0
Number of Members	0
Language1	0
Language2	290
Language3	351
Language4	356
Multiple_Language	0
National_Language_Used	0
EU	0
NATO	0
Country_Group	0
MyESB_Community	0
MyESB_Personal	0

OGAE_Points	0
Qualification_Record	46

dtype: int64

Data Collection, Inspection and Cleaning-lyrics dataframe

Data Collection

```
import kagglehub

# Download latest version dataset of lyrics
path = kagglehub.dataset_download("minitree/eurovision-song-lyrics")

print("Path to dataset files:", path)
```

```
Downloading from https://www.kaggle.com/api/v1/datasets/download/minitree/e
100%|██████████| 1.02M/1.02M [00:00<00:00, 39.7MB/s]Extracting files...
Path to dataset files: /root/.cache/kagglehub/datasets/minitree/eurovision-
```

```
import json
import pandas as pd

# Read in the lyrics file, please note that the downloading path might be diffe
json_file_path = f"/root/.cache/kagglehub/datasets/minitree/eurovision-song-lyı
with open(json_file_path, encoding="utf-8") as f:
    lyrics_raw = json.load(f)
```

Data Inspection

```
# Normalize the json file, and create a Dataframe
lyrics = pd.DataFrame.from_dict(lyrics_raw, orient='index')
```

```
# Inspect the head of lyrics df
lyrics.head()
```

	#	Country	#.1	Artist	Song	Language	Place	Score	Eurovision Number	Ye
0	1	Netherlands	1	Jetty Paerl	De vogels van Holland	Dutch	-	-	1	19
1	2	Switzerland	1	Lys Assia	Das alte Karussell	German	-	-	1	19
2	3	Belgium	1	Fud Leclerc	Messieurs les noyés de la Seine	French	-	-	1	19

Next steps: Generate code with lyrics | View recommended plots | New interactive sheet

```
# Inspect lyrics df info
lyrics.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1795 entries, 0 to 1794
Data columns (total 14 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   #                   1795 non-null   object
 1   Country             1795 non-null   object
 2   #.1                 1795 non-null   object
 3   Artist              1795 non-null   object
 4   Song                1795 non-null   object
 5   Language            1795 non-null   object
 6   Place               1795 non-null   object
 7   Score               1795 non-null   object
 8   Eurovision Number   1795 non-null   int64
 9   Year                1795 non-null   object
 10  Host Country        1795 non-null   object
 11  Host City           1795 non-null   object
 12  Lyrics              1795 non-null   object
 13  Lyrics translation  1795 non-null   object
dtypes: int64(1), object(13)
memory usage: 274.9+ KB
```

Data Cleaning

```
# Change the datatype of year to int
lyrics['Year'] = pd.to_numeric(lyrics['Year'], errors='coerce')

#Filter the songs from 2016-2025
lyrics_filtered = lyrics[lyrics['Year'].between(2016, 2025)]

# Inspect the sliced dataframe
lyrics_filtered.head()
```

	#	Country	#.1	Artist	Song	Language	Place	Score	Eurovis Num
1396	1397	Finland	50	Sandhja	Sing It Away	English	-	-	
1397	1398	Greece	37	Argo	Utopian Land	Greek/English (Pontic Greek)	-	-	
1398	1399	Moldova	12	Lidia Isac	Falling Stars	English	-	-	

Next steps: Generate code with lyrics_filtered | View recommended plots | New interactive sheet

```
# Normalize the language value
lyrics_filtered["Language"] = lyrics_filtered["Language"].str.strip()
```

```
<ipython-input-15-82c50204acca>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs
  lyrics_filtered["Language"] = lyrics_filtered["Language"].str.strip()
```

```
# Create a function that only when the language is English, then it takes the \
lyrics_filtered["Lyrics_Final"] = lyrics_filtered.apply(
    lambda row: row["Lyrics"]
    if row["Language"] == "English"
    else row["Lyrics translation"],
    axis=1
)
```

```
<ipython-input-16-157453347d81>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs
  lyrics_filtered["Lyrics_Final"] = lyrics_filtered.apply(
```

```
# Check for nulls or missing final lyrics
print(lyrics_filtered["Lyrics_Final"].isna().sum())
```

```
0
```

```
# Inspect the new column
lyrics_filtered.head()
```

	#	Country	#.1	Artist	Song	Language	Place	Score	Eurovis Num
1396	1397	Finland	50	Sandhja	Sing It Away	English	-	-	
1397	1398	Greece	37	Argo	Utopian Land	Greek/English (Pontic Greek)	-	-	
1398	1399	Moldova	12	Lidia Isac	Falling Stars	English	-	-	

Next steps: Generate code with `lyrics_filtered` View recommended plots New interactive sheet

```
# Install packages
! pip install pandas scikit-learn matplotlib seaborn bertopic
```

```
  Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-manylinux2014_x86_64.
```

```
Requirement already satisfied: triton==3.2.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.11/d
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.
Requirement already satisfied: tokenizers<0.22,>=0.21 in /usr/local/lib/pyt
Requirement already satisfied: safetensors>=0.4.3 in /usr/local/lib/python3
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/p
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/di
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3
Downloading bertopic-0.17.0-py3-none-any.whl (150 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 150.6/150.6 kB 5.2 MB/s eta 0:0
Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-manylinux2014_x86_64.whl (
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 363.4/363.4 MB 4.4 MB/s eta 0:0
Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-manylinux2014_x86_64.w
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.8/13.8 MB 71.6 MB/s eta 0:00
Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-manylinux2014_x86_64.w
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.6/24.6 MB 83.5 MB/s eta 0:00
Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-manylinux2014_x86_64
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 883.7/883.7 kB 58.7 MB/s eta 0:
Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-manylinux2014_x86_64.whl (6
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 664.8/664.8 MB 3.3 MB/s eta 0:0
Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-manylinux2014_x86_64.whl (2
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 211.5/211.5 MB 1.9 MB/s eta 0:0
Downloading nvidia_curand_cu12-10.3.5.147-py3-none-manylinux2014_x86_64.whl
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.3/56.3 MB 12.7 MB/s eta 0:00
Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-manylinux2014_x86_64.whl
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 127.9/127.9 MB 7.8 MB/s eta 0:0
Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-manylinux2014_x86_64.w
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.5/207.5 MB 5.8 MB/s eta 0:0
Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-manylinux2014_x86_64.wh
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.1/21.1 MB 71.4 MB/s eta 0:00
Installing collected packages: nvidia-nvjitlink-cu12, nvidia-curand-cu12, n
  Attempting uninstall: nvidia-nvjitlink-cu12
    Found existing installation: nvidia-nvjitlink-cu12 12.5.82
    Uninstalling nvidia-nvjitlink-cu12-12.5.82:
      Successfully uninstalled nvidia-nvjitlink-cu12-12.5.82
  Attempting uninstall: nvidia-curand-cu12
    Found existing installation: nvidia-curand-cu12 10.3.6.82
    Uninstalling nvidia-curand-cu12-10.3.6.82:
      Successfully uninstalled nvidia-curand-cu12-10.3.6.82
  Attempting uninstall: nvidia-cufft-cu12
    Found existing installation: nvidia-cufft-cu12 11.2.3.61
    Uninstalling nvidia-cufft-cu12-11.2.3.61:
      Successfully uninstalled nvidia-cufft-cu12-11.2.3.61
  Attempting uninstall: nvidia-cuda-runtime-cu12
    Found existing installation: nvidia-cuda-runtime-cu12 12.5.82
    Uninstalling nvidia-cuda-runtime-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-runtime-cu12-12.5.82
  Attempting uninstall: nvidia-cuda-nvrtc-cu12
    Found existing installation: nvidia-cuda-nvrtc-cu12 12.5.82
    Uninstalling nvidia-cuda-nvrtc-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-nvrtc-cu12-12.5.82
```

```
    Attempting uninstall: nvidia-cuda-cupti-cu12
      Found existing installation: nvidia-cuda-cupti-cu12 12.5.82
      Uninstalling nvidia-cuda-cupti-cu12-12.5.82:
        Successfully uninstalled nvidia-cuda-cupti-cu12-12.5.82
    Attempting uninstall: nvidia-cublas-cu12
```

Topic Modelling- Unsuperivised Machine Learning

LDA Model

```
# Import packages
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation
from bertopic import BERTopic
import matplotlib.pyplot as plt
import seaborn as sns


# Determine documents for LDA and BERTopic
documents = lyrics_filtered["Lyrics_Final"].tolist()


# Determine the vectorizer
vectorizer = CountVectorizer(stop_words="english", max_df=0.95, min_df=5)


# Fit the LDA model
# Turn our text data into a matrix of token counts
X = vectorizer.fit_transform(documents)
# Set up the LDA model to find 10 topics in the data
lda_model = LatentDirichletAllocation(n_components=10, random_state=42)
# Run LDA on our word count matrix — this gives us topic probabilities for eacl
lda_topics = lda_model.fit_transform(X)
# For each doc, grab the topic with the highest score (i.e., the one it’s mostl
lda_topic_assignments = lda_topics.argmax(axis=1)
# Add those topic labels back into our DataFrame so we can see what topic each
lyrics_filtered["LDA_Topic"] = lda_topic_assignments
```

```
# Inspect the LDA topic
lyrics_filtered.head()
```

	#	Country	#.1	Artist	Song	Language	Place	Score	Eurovis Num
1396	1397	Finland	50	Sandhja	Sing It Away	English	-	-	
1397	1398	Greece	37	Argo	Utopian Land	Greek/English (Pontic Greek)	-	-	
1398	1399	Moldova	12	Lidia Isac	Falling Stars	English	-	-	

Next steps: Generate code with lyrics_filtered | View recommended plots | New interactive sheet

```
def print_lda_topics(lda_model, vectorizer, n_top_words=10):
  # Get all the actual words (features) the model learned from
    feature_names = vectorizer.get_feature_names_out()
    # Loop through each topic found by the LDA model
    for idx, topic in enumerate(lda_model.components_):
       # Get the top N words for this topic (the ones with the highest weight)
        top_words = [feature_names[i] for i in topic.argsort()[:-n_top_words -
         # Print the topic number and its top words
        print(f"Topic {idx}: {' | '.join(top_words)}")
```

```
# Show the top words for each topic in the trained LDA model
print_lda_topics(lda_model, vectorizer)
```

```
Topic 0: love | ll | chorus | yeah | way | gonna | light | verse | tonight
Topic 1: la | like | falling | heart | life | sun | blood | just | hold | v
Topic 2: na | know | ll | life | hey | say | like | high | sound | hope
Topic 3: oh | just | don | pa | feel | chorus | good | cause | wanna | bigg
Topic 4: ooh | love | heart | look | know | don | need | friend | cause | l
Topic 5: like | gonna | baby | know | don | ll | tell | time | going | ya
Topic 6: love | don | got | ain | know | chorus | like | cause | come | sca
Topic 7: ah | ich | walking | chorus | holding | water | ve | die | fly | r
Topic 8: away | run | sing | right | night | make | come | far | land | gon
Topic 9: let | don | ll | like | want | know | dance | say | feel | hear
```

```
# I manually put a topic label for each topic, map the label here
topic_labels = {
    0: "Romantic/Intimate",
    1: "Emotional/Inspirational",
    2: "Energetic/Upbeat Chant",
    3: "Fun/Party Anthem",
    4: "Love & Friendship",
    5: "Relationship Conflict",
    6: "Vulnerability/Overcoming",
    7: "Dreamy/Abstract Imagery",
    8: "Journey/Escape",
    9: "Celebration/Desire"
}

# Apply to DataFrame
lyrics_filtered["LDA_Topic_Label"] = lyrics_filtered["LDA_Topic"].map(topic_lat
```

```
# Inspect the LDA topic label
lyrics_filtered.head()
```

	#	Country	#.1	Artist	Song	Language	Place	Score	Eurovis Num
1396	1397	Finland	50	Sandhja	Sing It Away	English	-	-	
1397	1398	Greece	37	Argo	Utopian Land	Greek/English (Pontic Greek)	-	-	
1398	1399	Moldova	12	Lidia Isac	Falling Stars	English	-	-	
1399	1400	Hungary	14	Freddie	Pioneer	English	19	108	

Next steps: Generate code with lyrics_filtered | View recommended plots | New interactive sheet

BERT Topic

```
# Dropping rows where "Lyrics_Final" is NaN
lyrics_df = lyrics_filtered.dropna(subset=["Lyrics_Final"])
# Excluding rows where "Lyrics_Final" is "english", empty, "none", or "nan"
lyrics_df = lyrics_df[~lyrics_df["Lyrics_Final"].str.strip().str.lower().isin([


# Keep only rows where the lyrics have more than 3 words
lyrics_df = lyrics_df[lyrics_df["Lyrics_Final"].str.split().str.len() > 3]


import re
import nltk
from nltk.corpus import stopwords
# Download stopwords
try:
    stopwords.words('english')
except LookupError:
    nltk.download('stopwords')

# Load English stop words into a set for faster lookup
ENGLISH_STOP_WORDS = set(stopwords.words('english'))

# Function to clean text: remove newlines, non-letter characters, lowercase, ar
def clean_text(text):
    text = re.sub(r"\n", " ", text)
    text = re.sub(r"[^a-zA-Z ]", "", text)
    words = text.lower().split()
    return " ".join([w for w in words if w not in ENGLISH_STOP_WORDS])

# Apply cleaning function to lyrics
lyrics_df["Lyrics_Cleaned"] = lyrics_df["Lyrics_Final"].apply(clean_text)
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
from sentence_transformers import SentenceTransformer
# Convert cleaned lyrics into a list of documents
docs = lyrics_df["Lyrics_Cleaned"].tolist()

# Load a pre-trained transformer model
embedding_model = SentenceTransformer("all-MiniLM-L6-v2")

# Generate sentence embeddings for each document with a progress bar
embeddings = embedding_model.encode(docs, show_progress_bar=True)
```

```
modules.json: 100%    349/349 [00:00<00:00, 5.09kB/s]
config_sentence_transformers.json: 100%    116/116 [00:00<00:00, 2.92kB/s]
README.md: 100%    10.5k/10.5k [00:00<00:00, 476kB/s]
sentence_bert_config.json: 100%    53.0/53.0 [00:00<00:00, 906B/s]
config.json: 100%    612/612 [00:00<00:00, 14.2kB/s]
Xet Storage is enabled for this repo, but the 'hf_xet' package is not insta
WARNING:huggingface_hub.file_download:Xet Storage is enabled for this repo,
model.safetensors: 100%    90.9M/90.9M [00:01<00:00, 101MB/s]
tokenizer_config.json: 100%    350/350 [00:00<00:00, 6.45kB/s]
vocab.txt: 100%    232k/232k [00:00<00:00, 2.80MB/s]
tokenizer.json: 100%    466k/466k [00:00<00:00, 5.49MB/s]
special_tokens_map.json: 100%    112/112 [00:00<00:00, 6.03kB/s]
config.json: 100%    190/190 [00:00<00:00, 4.62kB/s]
Batches: 100%    12/12 [00:34<00:00,  1.68s/it]
```

```
from sklearn.cluster import KMeans
from sentence_transformers import SentenceTransformer
from bertopic import BERTopic

# Initialize KMeans clustering with a fixed number of clusters (topics)
kmeans_model = KMeans(n_clusters=10, random_state=42)

# Initialize BERTopic with precomputed embedding model and KMeans clustering
bertopic_model = BERTopic(embedding_model=embedding_model, hdbscan_model=kmeans

# Fit the BERTopic model on the documents and their embeddings to extract topic
topics, probs = bertopic_model.fit_transform(docs, embeddings)


# Assign the predicted BERTopic topic for each document to the DataFrame
lyrics_df["BERT_Topic"] = topics


# Extract the info of BERTopic
bertopic_model.get_topic_info()
```

	Topic	Count	Name	Representation	Representative_Docs
0	0	63	0_im_love_dont_heart	[im, love, dont, heart, let, know, time, never...	[look used rockstars never thought harm til th...
1	1	50	1_sowing_like_love_lover	[sowing, like, love, lover, night, sun, come, ...	[oh spring song spring song spent winter garde...
2	2	42	2_bam_ea_ich_one	[bam, ea, ich, one, ey, love, ill, take, ya, n...	[verse want stay tonight far every sight every...
3	3	40	3_oh_chorus_verse_im	[oh, chorus, verse, im, sing, dont, away, gonn...	[verse jessika bullied moment born always one ...
4	4	39	4_im_go_yeah_gonna	[im, go, yeah, gonna, oh, wanna, yay, like, go...	[verse see look eyes aint feeling pressure pre...

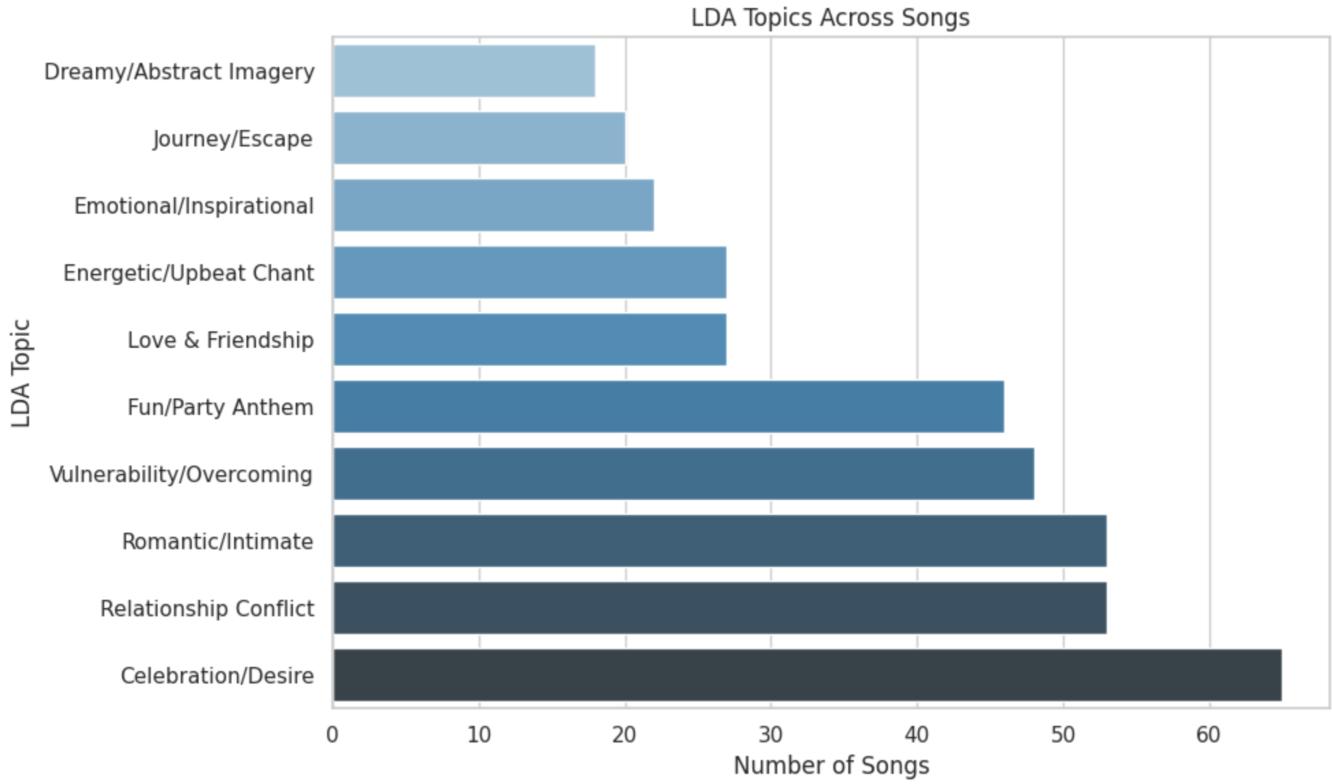
```
# Asign the topics in a DataFrame
topic_info = bertopic_model.get_topic_info()
```

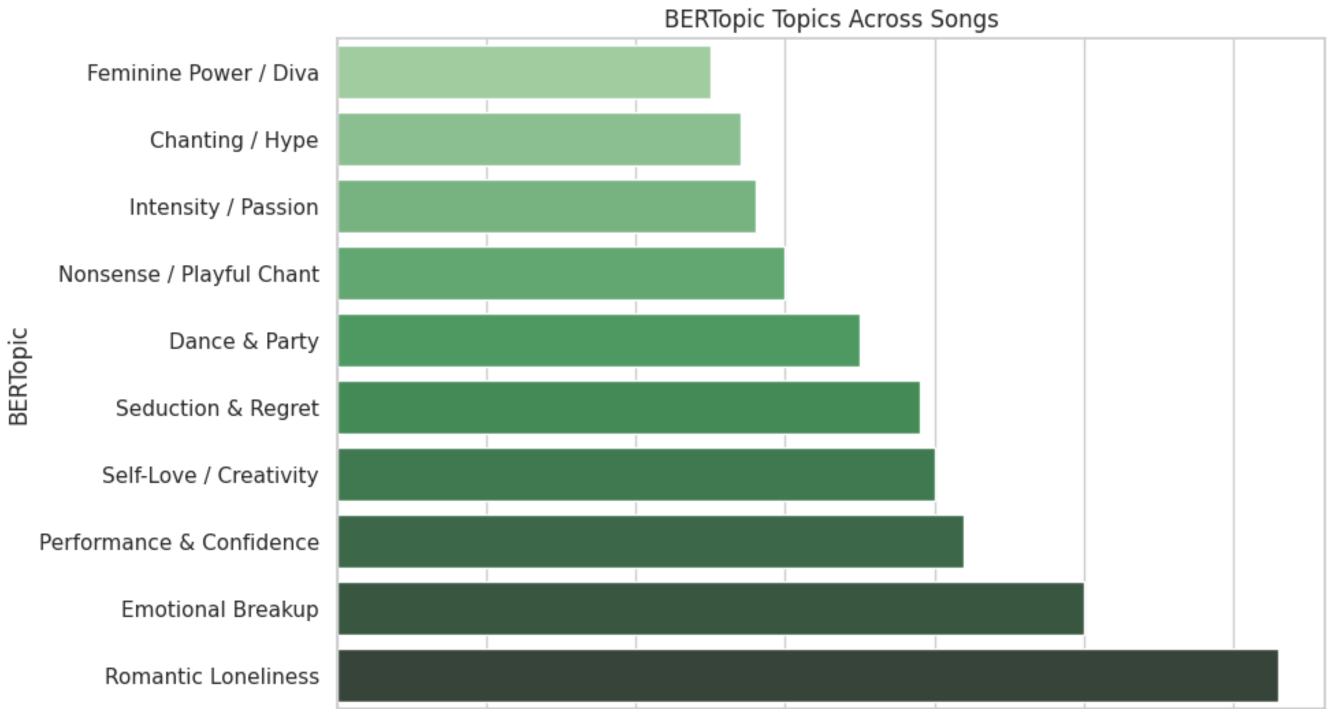
```
import pandas as pd

# Show all rows and all columns
pd.set_option("display.max_rows", None)
pd.set_option("display.max_columns", None)
pd.set_option("display.max_colwidth", None)

# Print the full DataFrame
print(topic_info)
```

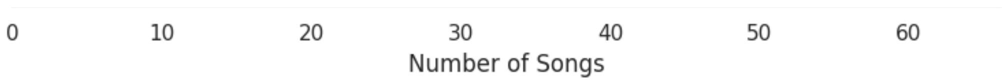
```
   Topic  Count                        Name  \
0      0     63       0_im_love_dont_heart
1      1     50  1_sowing_like_love_lover
2      2     42          2_bam_ea_ich_one
3      3     40       3_oh_chorus_verse_im
4      4     39         4_im_go_yeah_gonna
5      5     35          5_la_im_diva_know
6      6     30   6_love_ill_italy_wasted
7      7     28      7_love_oh_bigger_feel
8      8     27      8_sauna_im_dont_fire
9      9     25          9_poe_na_yum_cha

                                                        Representation  \
0          [im, love, dont, heart, let, know, time, never, go, like]
1      [sowing, like, love, lover, night, sun, come, go, good, never]
2                 [bam, ea, ich, one, ey, love, ill, take, ya, never]
3      [oh, chorus, verse, im, sing, dont, away, gonna, hear, friend]
4             [im, go, yeah, gonna, oh, wanna, yay, like, got, dance]
5   [la, im, diva, know, like, dont, healthy, sleep, rules, supergirl]
6          [love, ill, italy, wasted, dont, time, us, im, one, kiss]
7  [love, oh, bigger, feel, chorus, waiting, verse, alive, make, life]
8  [sauna, im, dont, fire, burns, feel, survivor, burning, cant, like]
9              [poe, na, yum, cha, mamma, im, tim, freaky, like, rim]



0
1
2
3
4
5
6
7
8
9  [ohoohoohoohoohooh sure told really like teeth hairy coat nothing undern
```

```
# I manually put a topic label for each topic, map the label here

topic_labels = {
    0: "Romantic Loneliness",
    1: "Emotional Breakup",
    2: "Performance & Confidence",
    3: "Self-Love / Creativity",
    4: "Seduction & Regret",
    5: "Dance & Party",
    6: "Nonsense / Playful Chant",
    7: "Intensity / Passion",
    8: "Chanting / Hype",
    9: "Feminine Power / Diva"
}


# Map numerical BERT topic IDs to descriptive labels
lyrics_df["BERT_Topic_Label"] = lyrics_df["BERT_Topic"].map(topic_labels)


import seaborn as sns
import matplotlib.pyplot as plt

# Set style and figure size
sns.set(style="whitegrid")
plt.figure(figsize=(10, 6))

# Sort topics by count
lda_order = lyrics_df["LDA_Topic_Label"].value_counts().sort_values().index

# LDA plot: Horizontal bars for clarity
sns.countplot(
    y="LDA_Topic_Label",
    data=lyrics_df,
    order=lda_order,
    palette="Blues_d"
)
plt.title("LDA Topics Across Songs")
plt.xlabel("Number of Songs")
plt.ylabel("LDA Topic")
plt.tight_layout()
plt.show()

# BERTopic plot: similar chart
plt.figure(figsize=(10, 6))
bert_order = lyrics_df["BERT_Topic_Label"].value_counts().sort_values().index

sns.countplot(
```

```
    y="BERT_Topic_Label",
    data=lyrics_df,
    order=bert_order,
    palette="Greens_d"
)
plt.title("BERTopic Topics Across Songs")
plt.xlabel("Number of Songs")
plt.ylabel("BERTopic")
plt.tight_layout()
plt.show()
```







Statistical Testing and Supervised Machine Learning Model Selection

Merging

```
# Inspect the column names for merging
print(lyrics_df.columns)
print(result.columns)
```

```
Index(['#', 'Country', '#.1', 'Artist', 'Song', 'Language', 'Place', 'Score
       'Eurovision Number', 'Year', 'Host Country', 'Host City', 'Lyrics',
       'Lyrics translation', 'Lyrics_Final', 'LDA_Topic', 'LDA_Topic_Label'
       'Lyrics_Cleaned', 'BERT_Topic', 'BERT_Topic_Label'],
      dtype='object')
Index(['Year', 'Country ', 'Song ', 'Artist ', 'Final_Place', 'Final_Points
       'Top 5', 'Top 10', 'Running_Order_Final', 'Grand_Final_Ind', 'Big6_I
       'Semi_Final_Num', 'Semi_Place', 'Semi_Points', 'Running_Order_Semi',
       'National_Final', 'Solo_Artist', 'Sex', 'Returning_Artist_Ind',
       'Number of Members', 'Language1', 'Language2', 'Language3', 'Languag
       'Multiple_Language', 'National_Language_Used', 'EU', 'NATO',
       'Country_Group', 'MyESB_Community', 'MyESB_Personal', 'OGAE_Points',
       'Qualification_Record'],
      dtype='object')
```

```
import re
import unicodedata

# Function to normalize and clean country names
def clean_country(c):
    c = str(c).lower() # Convert to lowercase
    c = re.sub(r"\(.*?\)", "", c)  # Remove (2), etc.
    c = re.sub(r"[^a-z ]", "", c).strip() # Remove non-alphabetic characters
    return c

# Apply cleaning function to prepare country names for merging
lyrics_df["Country_Merge"] = lyrics_df["Country"].apply(clean_country)
result["Country_Merge"] = result["Country "].apply(clean_country)
```

```
# Check the length of two Dataframe for merging
print("Lyrics rows:", len(lyrics_df))
print("Result rows:", len(result))
```

```
Lyrics rows: 379
Result rows: 358
```

```
# Check the length of two Dataframe for merging
result['Country_Merge'].value_counts()
```

	count
Country_Merge	

albania	9
australia	9
azerbaijan	9
austria	9
belgium	9
croatia	9
estonia	9
cyprus	9
czech republic	9
denmark	9
france	9
finland	9
georgia	9
germany	9
switzerland	9
greece	9
iceland	9
ireland	9
israel	9
italy	9
latvia	9
lithuania	9
malta	9
netherlands	9
poland	9
norway	9
san marino	9
spain	9
slovenia	9
serbia	9

sweden	9
united kingdom	9
armenia	8
portugal	8
ukraine	8
moldova	8
romania	6
montenegro	6
bulgaria	5
hungary	4
russia	4
belarus	4
fyr macedonia	3
north macedonia	3
luxembourg	2
bosnia herzegovina	1

dtype: int64

```
# Check the length of two Dataframe for merging
lyrics_df['Country_Merge'].value_counts()
```

	count
Country_Merge	
finland	10
greece	10
austria	10
estonia	10
iceland	10
united kingdom	10
sweden	10

albania	10
slovenia	10
australia	10
georgia	10
serbia	10
ireland	10
switzerland	10
poland	10
latvia	10
germany	10
belgium	10
italy	10
spain	10
lithuania	9
malta	9
france	9
azerbaijan	9
san marino	9
moldova	9
armenia	9
portugal	9
norway	9
cyprus	8
croatia	8
ukraine	8
denmark	8
netherlands	7
romania	7
israel	7
montenegro	6

czech republic	6
bulgaria	5
belarus	5
hungary	4
russia	4
north macedonia	4
macedonia	3
czechia	3
the netherlands	2
luxembourg	2
bosnia and herzegovina	1

dtype: int64

```
# Merge two datasets
merged_df = pd.merge(
    lyrics_df,
    result,
    on=["Year", "Country_Merge"],
    how="right"
)
print("Matched entries:", len(merged_df))
```

```
Matched entries: 358
```

```
# Inspect merged DataFrame
merged_df.head()
```

#	Country	#.1	Artist	Song	Language	Place	Score	Eurovisi Numb

0	1759	Albania	-	Shkodra Elektronike	Zjerm	Albanian	-	-	69

1	1760	Armenia	-	PARG	SURVIVOR	English	-	-	69

2	1761	Australia	-	Go-Jo	Milkshake Man	English	-	-	6(

3	1762	Austria	-	JJ	Wasted Love	English	-	-	69

4	1763	Azerbaijan	-	Mamagama	Run With U	English	-	-	69

```
# Check the columns
print(merged_df.columns)
```

```
Index(['#', 'Country', '#.1', 'Artist', 'Song', 'Language', 'Place', 'Score
       'Eurovision Number', 'Year', 'Host Country', 'Host City', 'Lyrics',
       'Lyrics translation', 'Lyrics_Final', 'LDA_Topic', 'LDA_Topic_Label'
       'Lyrics_Cleaned', 'BERT_Topic', 'BERT_Topic_Label', 'Country_Merge',
       'Country ', 'Song ', 'Artist ', 'Final_Place', 'Final_Points', 'Top
       'Top 10', 'Running_Order_Final', 'Grand_Final_Ind', 'Big6_Ind',
       'Semi_Final_Num', 'Semi_Place', 'Semi_Points', 'Running_Order_Semi',
       'National_Final', 'Solo_Artist', 'Sex', 'Returning_Artist_Ind',
       'Number of Members', 'Language1', 'Language2', 'Language3', 'Languag
       'Multiple_Language', 'National_Language_Used', 'EU', 'NATO',
       'Country_Group', 'MyESB_Community', 'MyESB_Personal', 'OGAE_Points',
       'Qualification_Record'],
      dtype='object')
```

Exploratory Analysis

```
# Compute the average Final_Place for each LDA topic, sorted from worst (highes
merged_df.groupby("LDA_Topic_Label")["Final_Place"].mean().sort_values(ascendir
```

LDA_Topic_Label	Final_Place
Romantic/Intimate	15.096774
Fun/Party Anthem	14.552632
Energetic/Upbeat Chant	14.545455
Celebration/Desire	14.500000
Love & Friendship	14.411765
Vulnerability/Overcoming	13.695652
Emotional/Inspirational	13.000000
Journey/Escape	12.250000
Relationship Conflict	11.225806
Dreamy/Abstract Imagery	10.000000

dtype: float64

```
# Compute the average Final_Points for each LDA topic, sorted from worst (highe
merged_df.groupby("LDA_Topic_Label")["Final_Points"].mean().sort_values(ascendi
```

LDA_Topic_Label	Final_Points
Dreamy/Abstract Imagery	242.166667
Relationship Conflict	232.129032
Vulnerability/Overcoming	197.434783
Emotional/Inspirational	164.818182
Energetic/Upbeat Chant	164.636364
Journey/Escape	164.000000
Fun/Party Anthem	158.868421
Celebration/Desire	155.031250
Love & Friendship	147.470588
Romantic/Intimate	146.903226

dtype: float64

```
# Compute the average probability to get intio final for each LDA topic, sortec
merged_df.groupby("LDA_Topic_Label")["Grand_Final_Ind"].mean().sort_values(asce
```

LDA_Topic_Label	Grand_Final_Ind
Fun/Party Anthem	0.866667
Love & Friendship	0.708333
Dreamy/Abstract Imagery	0.705882
Relationship Conflict	0.673913
Romantic/Intimate	0.659574
Celebration/Desire	0.615385
Emotional/Inspirational	0.611111
Vulnerability/Overcoming	0.547619
Energetic/Upbeat Chant	0.500000
Journey/Escape	0.421053

dtype: float64

```
# Compute the average probability to get intio Top5 for each LDA topic, sorted
merged_df.groupby("LDA_Topic_Label")["Top 5"].mean().sort_values(ascending=Fals
```

LDA_Topic_Label	Top 5
Relationship Conflict	0.222222
Vulnerability/Overcoming	0.151515
Dreamy/Abstract Imagery	0.125000
Fun/Party Anthem	0.108108
Celebration/Desire	0.090909
Love & Friendship	0.047619
Romantic/Intimate	0.023256
Journey/Escape	0.000000
Emotional/Inspirational	0.000000
Energetic/Upbeat Chant	0.000000

dtype: float64

```
# Compute the average probability to get intio Top 10 for each LDA topic, sorte
merged_df.groupby("LDA_Topic_Label")["Top 10"].mean().sort_values(ascending=Fal
```

LDA_Topic_Label	Top 10
Relationship Conflict	0.394737
Dreamy/Abstract Imagery	0.235294
Celebration/Desire	0.222222
Fun/Party Anthem	0.216216
Vulnerability/Overcoming	0.151515
Emotional/Inspirational	0.125000
Love & Friendship	0.095238
Romantic/Intimate	0.068182
Journey/Escape	0.000000
Energetic/Upbeat Chant	0.000000

dtype: float64

```
# Compute the average Semi final place for each LDA topic, sorted from worst (h
merged_df.groupby("LDA_Topic_Label")["Semi_Place"].mean().sort_values(ascending
```

LDA_Topic_Label	Semi_Place
Journey/Escape	12.562500
Energetic/Upbeat Chant	10.333333
Vulnerability/Overcoming	10.250000
Romantic/Intimate	9.837209
Relationship Conflict	9.062500
Celebration/Desire	9.021739
Emotional/Inspirational	8.500000
Love & Friendship	8.238095
Dreamy/Abstract Imagery	7.230769
Fun/Party Anthem	6.694444

dtype: float64

```
# Compute the average Semi final points for each LDA topic, sorted from worst (
merged_df.groupby("LDA_Topic_Label")["Semi_Points"].mean().sort_values(ascendir
```

LDA_Topic_Label	Semi_Points
Dreamy/Abstract Imagery	146.769231
Fun/Party Anthem	132.500000
Love & Friendship	122.000000
Romantic/Intimate	115.255814
Relationship Conflict	113.906250
Emotional/Inspirational	113.625000
Celebration/Desire	110.260870
Energetic/Upbeat Chant	106.047619
Vulnerability/Overcoming	99.416667
Journey/Escape	68.875000

dtype: float64

```
# Compute the average probability to get intio Top5 for each BERT topic, sortec
merged_df.groupby("BERT_Topic_Label")["Top 5"].mean().sort_values(ascending=Fal
```

BERT_Topic_Label	Top 5
Feminine Power / Diva	0.285714
Nonsense / Playful Chant	0.277778
Dance & Party	0.222222
Emotional Breakup	0.114286
Chanting / Hype	0.111111
Romantic Loneliness	0.052632
Seduction & Regret	0.034483
Performance & Confidence	0.027778
Intensity / Passion	0.000000
Self-Love / Creativity	0.000000

dtype: float64

```
# Compute the average probability to get intio Top10 for each BERT topic, sorte
merged_df.groupby("BERT_Topic_Label")["Top 10"].mean().sort_values(ascending=Fa
```

	Top 10
BERT_Topic_Label	
Nonsense / Playful Chant	0.444444
Feminine Power / Diva	0.400000
Dance & Party	0.310345
Chanting / Hype	0.263158
Romantic Loneliness	0.210526
Emotional Breakup	0.194444
Performance & Confidence	0.083333
Seduction & Regret	0.068966
Self-Love / Creativity	0.026316
Intensity / Passion	0.000000

dtype: float64

```
# Compute the average Final_Points for each BERT topic, sorted from worst (high
merged_df.groupby("BERT_Topic_Label")["Final_Points"].mean().sort_values(ascend
```

BERT_Topic_Label	Final_Points
Feminine Power / Diva	265.600000
Nonsense / Playful Chant	223.058824
Seduction & Regret	188.150000
Performance & Confidence	181.500000
Intensity / Passion	175.937500
Dance & Party	172.920000
Romantic Loneliness	168.750000
Self-Love / Creativity	162.555556
Emotional Breakup	145.111111
Chanting / Hype	126.761905

dtype: float64

```
# Compute the average Final_Place for each BERT topic, sorted from worst (highe
merged_df.groupby("BERT_Topic_Label")["Final_Place"].mean().sort_values(ascendi
```

BERT_Topic_Label	Final_Place
Chanting / Hype	16.047619
Emotional Breakup	15.407407
Self-Love / Creativity	14.592593
Intensity / Passion	14.187500
Romantic Loneliness	14.071429
Nonsense / Playful Chant	12.882353
Dance & Party	12.600000
Performance & Confidence	12.444444
Seduction & Regret	12.300000
Feminine Power / Diva	9.466667

dtype: float64

```
# Compute the average probability to get intio final for each BERT topic, sorte
merged_df.groupby("BERT_Topic_Label")["Grand_Final_Ind"].mean().sort_values(asc
```

BERT_Topic_Label	Grand_Final_Ind
Chanting / Hype	0.875000
Nonsense / Playful Chant	0.782609
Dance & Party	0.781250
Self-Love / Creativity	0.692308
Feminine Power / Diva	0.681818
Seduction & Regret	0.625000
Emotional Breakup	0.613636
Romantic Loneliness	0.595745
Intensity / Passion	0.592593
Performance & Confidence	0.428571

dtype: float64

```
# Compute the average Semi final place for each BERT topic, sorted from worst (
merged_df.groupby("BERT_Topic_Label")["Semi_Place"].mean().sort_values(ascendir
```

BERT_Topic_Label	Semi_Place
Performance & Confidence	10.333333
Emotional Breakup	10.147059
Intensity / Passion	9.640000
Seduction & Regret	9.322581
Romantic Loneliness	9.300000
Self-Love / Creativity	8.406250
Feminine Power / Diva	8.375000
Chanting / Hype	8.315789
Dance & Party	8.083333
Nonsense / Playful Chant	7.700000

dtype: float64

```
# Compute the average Semi final points for each BERT topic, sorted from worst
merged_df.groupby("BERT_Topic_Label")["Semi_Points"].mean().sort_values(ascendi
```

BERT_Topic_Label	Semi_Points
Self-Love / Creativity	141.625000
Intensity / Passion	134.440000
Nonsense / Playful Chant	129.350000
Seduction & Regret	121.709677
Romantic Loneliness	107.650000
Dance & Party	102.625000
Feminine Power / Diva	101.625000
Performance & Confidence	100.025641
Emotional Breakup	96.558824
Chanting / Hype	95.105263

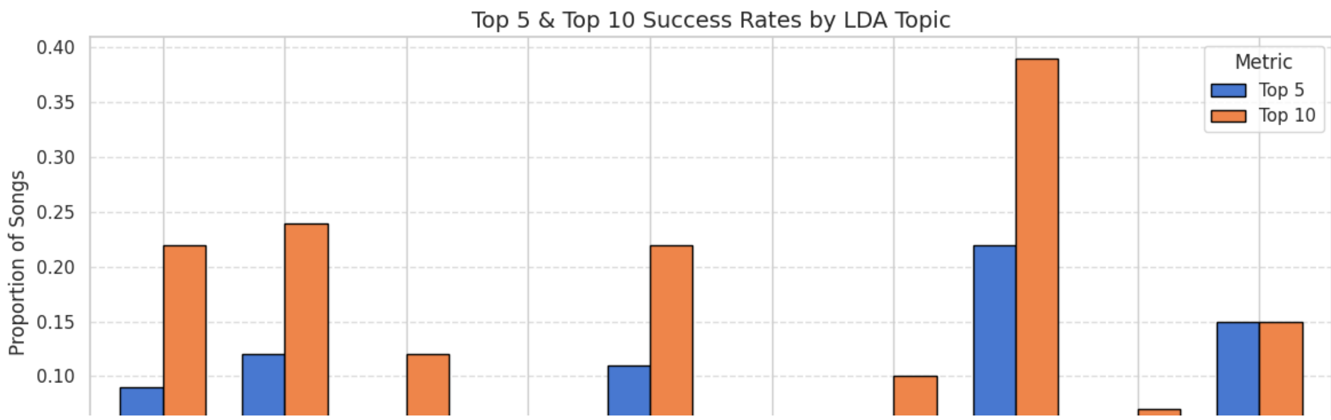
dtype: float64

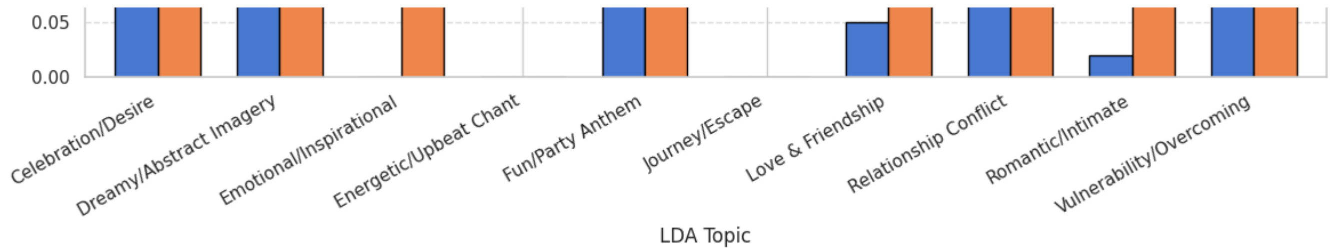
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Define the metrics of interest
metrics = [
    "Final_Place", "Final_Points", "Top 5", "Top 10",
    "Grand_Final_Ind", "Semi_Place", "Semi_Points"
]

# LDA: Group and summarize
lda_summary = merged_df.groupby("LDA_Topic_Label")[metrics].mean().round(2)

# BERT: Group and summarize
bert_summary = merged_df.groupby("BERT_Topic_Label")[metrics].mean().round(2)

# Display summary tables
print("=== LDA Topic Summary ===")
print(lda_summary)
print("\n=== BERTopic Summary ===")
print(bert_summary)
```

```
# Set a clean style
sns.set(style="whitegrid", palette="muted")

# === LDA Plot ===
plt.figure(figsize=(12, 6))
lda_plot = lda_summary[["Top 5", "Top 10"]].plot(
    kind="bar",
    figsize=(12, 6),
    width=0.7,
    edgecolor="black"
)
plt.title("Top 5 & Top 10 Success Rates by LDA Topic", fontsize=14)
plt.ylabel("Proportion of Songs", fontsize=12)
plt.xlabel("LDA Topic", fontsize=12)
plt.xticks(rotation=30, ha='right')
plt.tight_layout()
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.legend(title="Metric")
plt.show()

# BERTopic Plot
plt.figure(figsize=(12, 6))
bert_plot = bert_summary[["Top 5", "Top 10"]].plot(
    kind="bar",
    figsize=(12, 6),
    width=0.7,
    edgecolor="black"
)
plt.title("Top 5 & Top 10 Success Rates by BERTopic", fontsize=14)
plt.ylabel("Proportion of Songs", fontsize=12)
plt.xlabel("BERTopic", fontsize=12)
plt.xticks(rotation=30, ha='right')
plt.tight_layout()
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.legend(title="Metric")
plt.show()
```

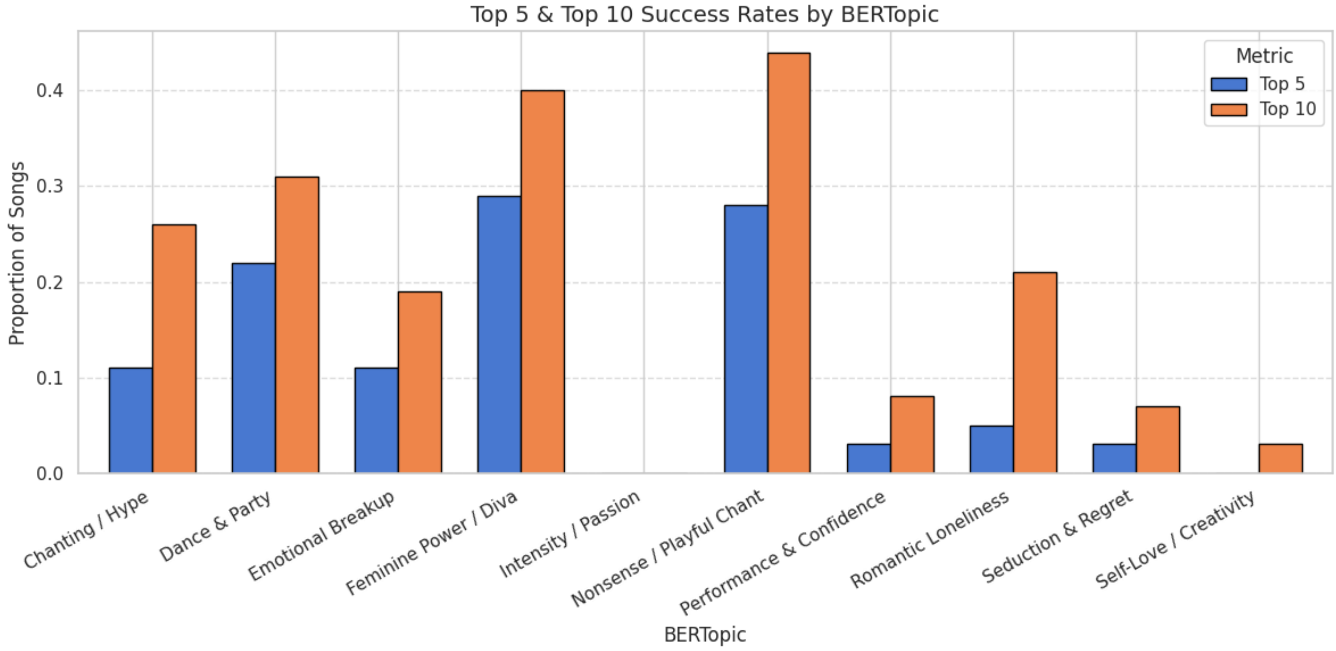
```
=== LDA Topic Summary ===
                            Final_Place  Final_Points  Top 5  Top 10  \
LDA_Topic_Label
Celebration/Desire                14.50        155.03   0.09    0.22
Dreamy/Abstract Imagery           10.00        242.17   0.12    0.24
Emotional/Inspirational           13.00        164.82   0.00    0.12
Energetic/Upbeat Chant            14.55        164.64   0.00    0.00
Fun/Party Anthem                  14.55        158.87   0.11    0.22
Journey/Escape                    12.25        164.00   0.00    0.00
Love & Friendship                 14.41        147.47   0.05    0.10
Relationship Conflict             11.23        232.13   0.22    0.39
Romantic/Intimate                 15.10        146.90   0.02    0.07
```

```
Romantic/Intimate              15.10        146.90   0.02    0.07
Vulnerability/Overcoming       13.70        197.43   0.15    0.15

                          Grand_Final_Ind  Semi_Place  Semi_Points
LDA_Topic_Label
Celebration/Desire                   0.62        9.02       110.26
Dreamy/Abstract Imagery              0.71        7.23       146.77
Emotional/Inspirational              0.61        8.50       113.62
Energetic/Upbeat Chant               0.50       10.33       106.05
Fun/Party Anthem                     0.87        6.69       132.50
Journey/Escape                       0.42       12.56        68.88
Love & Friendship                    0.71        8.24       122.00
Relationship Conflict                0.67        9.06       113.91
Romantic/Intimate                    0.66        9.84       115.26
Vulnerability/Overcoming             0.55       10.25        99.42

=== BERTopic Summary ===
                          Final_Place  Final_Points  Top 5  Top 10  \
BERT_Topic_Label
Chanting / Hype                 16.05        126.76   0.11    0.26
Dance & Party                   12.60        172.92   0.22    0.31
Emotional Breakup               15.41        145.11   0.11    0.19
Feminine Power / Diva            9.47        265.60   0.29    0.40
Intensity / Passion             14.19        175.94   0.00    0.00
Nonsense / Playful Chant        12.88        223.06   0.28    0.44
Performance & Confidence        12.44        181.50   0.03    0.08
Romantic Loneliness             14.07        168.75   0.05    0.21
Seduction & Regret              12.30        188.15   0.03    0.07
Self-Love / Creativity          14.59        162.56   0.00    0.03

                          Grand_Final_Ind  Semi_Place  Semi_Points
BERT_Topic_Label
Chanting / Hype                      0.88        8.32        95.11
Dance & Party                        0.78        8.08       102.62
Emotional Breakup                    0.61       10.15        96.56
Feminine Power / Diva                0.68        8.38       101.62
Intensity / Passion                  0.59        9.64       134.44
Nonsense / Playful Chant             0.78        7.70       129.35
Performance & Confidence             0.43       10.33       100.03
Romantic Loneliness                  0.60        9.30       107.65
Seduction & Regret                   0.62        9.32       121.71
Self-Love / Creativity               0.69        8.41       141.62
<Figure size 1200x600 with 0 Axes>
```






```
<Figure size 1200x600 with 0 Axes>
```



Satistical modelling

```
# One-hot encode LDA topics
lda_dummies = pd.get_dummies(merged_df["LDA_Topic_Label"], prefix="LDA")

# One-hot encode BERT topics
bert_dummies = pd.get_dummies(merged_df["BERT_Topic_Label"], prefix="BERT")

# Combine with base DataFrame
X_lda = pd.concat([merged_df[["Grand_Final_Ind", "Top 10", "Top 5"]], lda_dummi
X_bert = pd.concat([merged_df[["Grand_Final_Ind", "Top 10", "Top 5"]], bert_dum
```

```
# Fill 0 to the missing value
merged_df[["Top 5", "Top 10", "Grand_Final_Ind"]] = merged_df[["Top 5", "Top 10
```

```
print(merged_df["LDA_Topic_Label"].value_counts())
```

```
LDA_Topic_Label
Celebration/Desire           52
Romantic/Intimate            47
Relationship Conflict        46
Fun/Party Anthem             45
Vulnerability/Overcoming     42
Love & Friendship            24
Energetic/Upbeat Chant       22
Journey/Escape               19
Emotional/Inspirational      18
Dreamy/Abstract Imagery      17
Name: count, dtype: int64
```

```
import statsmodels.api as sm
import pandas as pd


# One-hot encode topics
X = pd.get_dummies(merged_df["LDA_Topic_Label"], prefix="LDA")
y = merged_df["Top 10"]

# Add constant and ensure float dtype
X = sm.add_constant(X).astype(float)

# Fit Generalized Linear Model with Binomial family
model = sm.GLM(y, X, family=sm.families.Binomial())
result = model.fit()

print(result.summary())
```

```
                 Generalized Linear Model Regression Results
==========================================================================
Dep. Variable:                 Top 10   No. Observations:
Model:                            GLM   Df Residuals:
Model Family:                Binomial   Df Model:
Link Function:                  Logit   Scale:                         1.0
Method:                          IRLS   Log-Likelihood:               -133
Date:                Wed, 28 May 2025   Deviance:                      267
Time:                        14:17:27   Pearson chi2:                    3
No. Iterations:                    22   Pseudo R-squ. (CS):           0.07
Covariance Type:            nonrobust
==========================================================================
                                   coef    std err          z      P>|z|
--------------------------------------------------------------------------
const                           -2.0369      0.614     -3.318      0.001
LDA_Celebration/Desire           0.6018      0.708      0.851      0.395
LDA_Dreamy/Abstract Imagery      0.8582      0.839      1.023      0.306
LDA_Emotional/Inspirational     -0.0426      0.969     -0.044      0.965
LDA_Energetic/Upbeat Chant     -21.5292   1.69e+04     -0.001      0.999
LDA_Fun/Party Anthem             0.5054      0.727      0.695      0.487
LDA_Journey/Escape             -21.5292   1.82e+04     -0.001      0.999
LDA_Love & Friendship           -0.3610      0.960     -0.376      0.707
LDA_Relationship Conflict        1.3109      0.690      1.901      0.057
LDA_Romantic/Intimate           -0.6487      0.856     -0.758      0.449
LDA_Vulnerability/Overcoming     0.0354      0.777      0.046      0.964
==========================================================================
```

```
# One-hot encode LDA topics
lda_dummies = pd.get_dummies(merged_df["LDA_Topic_Label"], prefix="LDA")

# Define predictors and outcome
X = lda_dummies
y = merged_df["Top 5"]

# Add constant and ensure float dtype
X = sm.add_constant(X).astype(float)

# Fit logistic regression model
model = sm.GLM(y, X, family=sm.families.Binomial())
result = model.fit()

# Print statistical summary
print(result.summary())
```

```
                 Generalized Linear Model Regression Results
==========================================================================
Dep. Variable:                  Top 5   No. Observations:
Model:                            GLM   Df Residuals:
Model Family:                Binomial   Df Model:
Link Function:                  Logit   Scale:                         1.0
Method:                          IRLS   Log-Likelihood:               -83.
Date:                Wed, 28 May 2025   Deviance:                      167
Time:                        14:17:27   Pearson chi2:                    2
No. Iterations:                    23   Pseudo R-squ. (CS):           0.05
Covariance Type:            nonrobust
==========================================================================
                                  coef    std err          z      P>|z|
--------------------------------------------------------------------------
const                          -3.2189      1.020     -3.156      0.002
LDA_Celebration/Desire          0.7340      1.145      0.641      0.521
LDA_Dreamy/Abstract Imagery     1.2040      1.268      0.950      0.342
LDA_Emotional/Inspirational   -21.3472   3.09e+04     -0.001      0.999
LDA_Energetic/Upbeat Chant    -21.3472   2.79e+04     -0.001      0.999
LDA_Fun/Party Anthem            0.8916      1.146      0.778      0.437
LDA_Journey/Escape            -21.3472   3.01e+04     -0.001      0.999
LDA_Love & Friendship           0.0834      1.443      0.058      0.954
LDA_Relationship Conflict       1.6607      1.091      1.522      0.128
LDA_Romantic/Intimate          -0.6098      1.436     -0.425      0.671
LDA_Vulnerability/Overcoming    1.2174      1.126      1.082      0.279
==========================================================================
```

```
# One-hot encode LDA topics
lda_dummies = pd.get_dummies(merged_df["LDA_Topic_Label"], prefix="LDA")

# Define predictors and outcome
X = lda_dummies
y = merged_df["Grand_Final_Ind"]

# Add constant and ensure float dtype
X = sm.add_constant(X).astype(float)

# Fit logistic regression model
model = sm.GLM(y, X, family=sm.families.Binomial())
result = model.fit()

# Print statistical summary
print(result.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:       Grand_Final_Ind   No. Observations:
Model:                           GLM   Df Residuals:
Model Family:               Binomial   Df Model:
Link Function:                 Logit   Scale:                          1.0
Method:                         IRLS   Log-Likelihood:                -221
Date:               Wed, 28 May 2025   Deviance:                       442
Time:                       14:17:27   Pearson chi2:                     3
No. Iterations:                    4   Pseudo R-squ. (CS):            0.05
Covariance Type:           nonrobust
==============================================================================
                                  coef    std err          z      P>|z|
------------------------------------------------------------------------------
const                           0.8109      0.425      1.908      0.056
LDA_Celebration/Desire         -0.3409      0.512     -0.666      0.505
LDA_Dreamy/Abstract Imagery     0.0645      0.681      0.095      0.925
LDA_Emotional/Inspirational    -0.3589      0.644     -0.558      0.577
LDA_Energetic/Upbeat Chant     -0.8109      0.602     -1.347      0.178
LDA_Fun/Party Anthem            1.0609      0.611      1.737      0.082
LDA_Journey/Escape             -1.1294      0.630     -1.794      0.073
LDA_Love & Friendship           0.0764      0.618      0.124      0.902
LDA_Relationship Conflict      -0.0850      0.529     -0.161      0.872
LDA_Romantic/Intimate          -0.1495      0.525     -0.285      0.776
LDA_Vulnerability/Overcoming   -0.6199      0.526     -1.178      0.239
==============================================================================
```

```
# One-hot encode LDA topics
bert_dummies = pd.get_dummies(merged_df["BERT_Topic_Label"], prefix="BERT")

# Define predictors and outcome
X = bert_dummies
y = merged_df["Top 10"]

# Add constant and ensure float dtype
X = sm.add_constant(X).astype(float)

# Fit logistic regression model
model = sm.GLM(y, X, family=sm.families.Binomial())
result = model.fit()

# Print statistical summary
print(result.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                 Top 10   No. Observations:
Model:                            GLM   Df Residuals:
Model Family:                Binomial   Df Model:
Link Function:                  Logit   Scale:                           1.0
Method:                          IRLS   Log-Likelihood:                 -132
Date:                Wed, 28 May 2025   Deviance:                        264
Time:                        14:17:27   Pearson chi2:                      3
No. Iterations:                    22   Pseudo R-squ. (CS):             0.08
Covariance Type:            nonrobust
==============================================================================
                                   coef    std err          z      P>|z|
------------------------------------------------------------------------------
const                           -2.0369      0.614     -3.318      0.001
BERT_Chanting / Hype             0.7019      0.793      0.885      0.376
BERT_Dance & Party               1.0986      0.729      1.507      0.132
BERT_Emotional Breakup           0.3719      0.739      0.503      0.615
BERT_Feminine Power / Diva       1.0561      0.778      1.357      0.175
BERT_Intensity / Passion       -21.5292   1.53e+04     -0.001      0.999
BERT_Nonsense / Playful Chant    1.4083      0.754      1.868      0.062
BERT_Performance & Confidence   -0.5281      0.858     -0.616      0.538
BERT_Romantic Loneliness         0.4528      0.726      0.623      0.533
BERT_Seduction & Regret         -0.6712      0.954     -0.704      0.482
BERT_Self-Love / Creativity     -1.6007      1.185     -1.351      0.177
==============================================================================
```

```
# One-hot encode LDA topics
bert_dummies = pd.get_dummies(merged_df["BERT_Topic_Label"], prefix="BERT")

# Define predictors and outcome
X = bert_dummies
y = merged_df["Top 5"]

# Add constant and ensure float dtype
X = sm.add_constant(X).astype(float)

# Fit logistic regression model
model = sm.GLM(y, X, family=sm.families.Binomial())
result = model.fit()

# Print statistical summary
print(result.summary())
```

```
                 Generalized Linear Model Regression Results
==========================================================================
Dep. Variable:                  Top 5   No. Observations:
Model:                            GLM   Df Residuals:
Model Family:                Binomial   Df Model:
Link Function:                  Logit   Scale:                         1.0
Method:                          IRLS   Log-Likelihood:               -79.
Date:                Wed, 28 May 2025   Deviance:                      159
Time:                        14:17:27   Pearson chi2:                    2
No. Iterations:                    23   Pseudo R-squ. (CS):           0.07
Covariance Type:            nonrobust
==========================================================================
                                   coef    std err          z      P>|z|
--------------------------------------------------------------------------
const                           -3.2189      1.020     -3.156      0.002
BERT_Chanting / Hype             0.8210      1.259      0.652      0.514
BERT_Dance & Party               1.7525      1.116      1.571      0.116
BERT_Emotional Breakup           0.9163      1.147      0.799      0.424
BERT_Feminine Power / Diva       1.7148      1.160      1.478      0.139
BERT_Intensity / Passion       -21.3472   2.52e+04     -0.001      0.999
BERT_Nonsense / Playful Chant    1.9379      1.138      1.703      0.089
BERT_Performance & Confidence   -0.4947      1.437     -0.344      0.731
BERT_Romantic Loneliness         0.1054      1.250      0.084      0.933
BERT_Seduction & Regret         -0.2151      1.440     -0.149      0.881
BERT_Self-Love / Creativity    -21.3472    2.1e+04     -0.001      0.999
==========================================================================
```

```
# One-hot encode LDA topics
bert_dummies = pd.get_dummies(merged_df["BERT_Topic_Label"], prefix="BERT")

# Define predictors and outcome
X = bert_dummies
y = merged_df["Grand_Final_Ind"]

# Add constant and ensure float dtype
X = sm.add_constant(X).astype(float)

# Fit logistic regression model
model = sm.GLM(y, X, family=sm.families.Binomial())
result = model.fit()

# Print statistical summary
print(result.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:        Grand_Final_Ind   No. Observations:
Model:                            GLM   Df Residuals:
Model Family:                Binomial   Df Model:
Link Function:                  Logit   Scale:                            1.0
Method:                          IRLS   Log-Likelihood:                  -220
Date:                Wed, 28 May 2025   Deviance:                         441
Time:                        14:17:27   Pearson chi2:                       3
No. Iterations:                     4   Pseudo R-squ. (CS):              0.05
Covariance Type:            nonrobust
==============================================================================
                                  coef    std err          z      P>|z|
------------------------------------------------------------------------------
const                           0.8109      0.425      1.908      0.056
BERT_Chanting / Hype            1.1350      0.749      1.515      0.130
BERT_Dance & Party              0.4620      0.603      0.766      0.443
BERT_Emotional Breakup         -0.3483      0.526     -0.662      0.508
BERT_Feminine Power / Diva     -0.0488      0.625     -0.078      0.938
BERT_Intensity / Passion       -0.4362      0.578     -0.755      0.450
BERT_Nonsense / Playful Chant   0.4700      0.660      0.712      0.477
BERT_Performance & Confidence  -1.0986      0.527     -2.084      0.037
BERT_Romantic Loneliness       -0.4232      0.519     -0.816      0.414
BERT_Seduction & Regret        -0.3001      0.560     -0.536      0.592
BERT_Self-Love / Creativity  2.078e-15      0.549   3.79e-15      1.000
==============================================================================
```

```
print(X.dtypes)
print(y.dtypes)
```

```
const                              float64
BERT_Chanting / Hype               float64
BERT_Dance & Party                 float64
BERT_Emotional Breakup             float64
BERT_Feminine Power / Diva         float64
BERT_Intensity / Passion           float64
BERT_Nonsense / Playful Chant      float64
BERT_Performance & Confidence      float64
BERT_Romantic Loneliness           float64
BERT_Seduction & Regret            float64
BERT_Self-Love / Creativity        float64
dtype: object
int64
```

```
import statsmodels.api as sm
import pandas as pd

# Prepare data: drop missing values
merged_df_clean = merged_df.dropna(subset=["Final_Points"])

# One-hot encode BERT topics
X = pd.get_dummies(merged_df_clean["LDA_Topic_Label"], prefix="LDA")
X = sm.add_constant(X)              # Add intercept term
X = X.astype(float)                 # Ensure numeric dtype

# Target variable
y = pd.to_numeric(merged_df_clean["Final_Points"], errors="coerce")

# Fit model
model = sm.OLS(y, X).fit()

# Show full statistical summary
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           Final_Points   R-squared:                       0.
Model:                            OLS   Adj. R-squared:                  0.
Method:                 Least Squares   F-statistic:                     1.
Date:                Wed, 28 May 2025   Prob (F-statistic):              0.
Time:                        14:17:28   Log-Likelihood:                -148
No. Observations:                 232   AIC:                             29
Df Residuals:                     221   BIC:                             30
Df Model:                          10
Covariance Type:            nonrobust
==============================================================================
                                  coef    std err          t      P>|t|
------------------------------------------------------------------------------
const                         225.7778     34.831      6.482      0.000
LDA_Celebration/Desire        -70.7465     43.539     -1.625      0.106
LDA_Dreamy/Abstract Imagery    16.3889     55.073      0.298      0.766
LDA_Emotional/Inspirational   -60.9596     56.555     -1.078      0.282
LDA_Energetic/Upbeat Chant    -61.1414     56.555     -1.081      0.281
LDA_Fun/Party Anthem          -66.9094     42.284     -1.582      0.115
LDA_Journey/Escape            -61.7778     62.793     -0.984      0.326
LDA_Love & Friendship         -78.3072     49.978     -1.567      0.119
LDA_Relationship Conflict       6.3513     43.791      0.145      0.885
LDA_Romantic/Intimate         -78.8746     43.791     -1.801      0.073
LDA_Vulnerability/Overcoming  -28.3430     46.505     -0.609      0.543
==============================================================================
Omnibus:                       48.490   Durbin-Watson:                   1.
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               73.
Skew:                           1.219   Prob(JB):                     9.65e
Kurtosis:                       4.298   Cond. No.                        1
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is corr
```

```python
# Prepare data: drop missing values
merged_df_clean = merged_df.dropna(subset=["Final_Place"])

# One-hot encode BERT topics
X = pd.get_dummies(merged_df_clean["LDA_Topic_Label"], prefix="LDA")
X = sm.add_constant(X)              # Add intercept term
X = X.astype(float)                 # Ensure numeric dtype

# Target variable
y = pd.to_numeric(merged_df_clean["Final_Place"], errors="coerce")

# Fit model
model = sm.OLS(y, X).fit()

# Show full statistical summary
```

```
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            Final_Place   R-squared:                       0.
Model:                            OLS   Adj. R-squared:                  0.
Method:                 Least Squares   F-statistic:                     1.
Date:                Wed, 28 May 2025   Prob (F-statistic):              0.
Time:                        14:17:28   Log-Likelihood:                -788
No. Observations:                 232   AIC:                             16
Df Residuals:                     221   BIC:                             16
Df Model:                          10
Covariance Type:            nonrobust
==============================================================================
                                   coef    std err          t      P>|t|
------------------------------------------------------------------------------
const                           10.7222      1.752      6.121      0.000
LDA_Celebration/Desire           3.7778      2.190      1.725      0.086
LDA_Dreamy/Abstract Imagery     -0.7222      2.770     -0.261      0.795
LDA_Emotional/Inspirational      2.2778      2.844      0.801      0.424
LDA_Energetic/Upbeat Chant       3.8232      2.844      1.344      0.180
LDA_Fun/Party Anthem             3.8304      2.127      1.801      0.073
LDA_Journey/Escape               1.5278      3.158      0.484      0.629
LDA_Love & Friendship            3.6895      2.514      1.468      0.144
LDA_Relationship Conflict        0.5036      2.202      0.229      0.819
LDA_Romantic/Intimate            4.3746      2.202      1.986      0.048
LDA_Vulnerability/Overcoming     2.9734      2.339      1.271      0.205
==============================================================================
Omnibus:                       43.498   Durbin-Watson:                   2.
Prob(Omnibus):                  0.000   Jarque-Bera (JB):                9.
Skew:                          -0.007   Prob(JB):                      0.00
Kurtosis:                       1.986   Cond. No.                         1
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is corr
```

```
# Prepare data: drop missing values
merged_df_clean = merged_df.dropna(subset=["Semi_Place"])

# One-hot encode BERT topics
X = pd.get_dummies(merged_df_clean["LDA_Topic_Label"], prefix="LDA")
X = sm.add_constant(X)              # Add intercept term
X = X.astype(float)                 # Ensure numeric dtype

# Target variable
y = pd.to_numeric(merged_df_clean["Semi_Place"], errors="coerce")

# Fit model
model = sm.OLS(y, X).fit()
```

```
# Show full statistical summary
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             Semi_Place   R-squared:                       0.
Model:                            OLS   Adj. R-squared:                  0.
Method:                 Least Squares   F-statistic:                     2.
Date:                Wed, 28 May 2025   Prob (F-statistic):            0.00
Time:                        14:17:28   Log-Likelihood:                -907
No. Observations:                 305   AIC:                             18
Df Residuals:                     294   BIC:                             18
Df Model:                          10
Covariance Type:            nonrobust
==============================================================================
                                  coef    std err          t      P>|t|
------------------------------------------------------------------------------
const                           7.6000      0.966      7.867      0.000
LDA_Celebration/Desire          1.4217      1.200      1.185      0.237
LDA_Dreamy/Abstract Imagery    -0.3692      1.652     -0.224      0.823
LDA_Emotional/Inspirational     0.9000      1.546      0.582      0.561
LDA_Energetic/Upbeat Chant      2.7333      1.430      1.912      0.057
LDA_Fun/Party Anthem           -0.9056      1.258     -0.720      0.472
LDA_Journey/Escape              4.9625      1.546      3.209      0.001
LDA_Love & Friendship           0.6381      1.430      0.446      0.656
LDA_Relationship Conflict       1.4625      1.289      1.134      0.258
LDA_Romantic/Intimate           2.2372      1.215      1.842      0.067
LDA_Vulnerability/Overcoming    2.6500      1.258      2.107      0.036
==============================================================================
Omnibus:                       57.401   Durbin-Watson:                   2.
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               13.
Skew:                           0.087   Prob(JB):                      0.00
Kurtosis:                       1.999   Cond. No.                         1
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is corr
```

```
# Prepare data: drop missing values
merged_df_clean = merged_df.dropna(subset=["Semi_Points"])

# One-hot encode BERT topics
X = pd.get_dummies(merged_df_clean["LDA_Topic_Label"], prefix="LDA")
X = sm.add_constant(X)                # Add intercept term
X = X.astype(float)                   # Ensure numeric dtype

# Target variable
y = pd.to_numeric(merged_df_clean["Semi_Points"], errors="coerce")

# Fit model
model = sm.OLS(y, X).fit()
```

```
# Show full statistical summary
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            Semi_Points   R-squared:                       0.
Model:                            OLS   Adj. R-squared:                  0.
Method:                 Least Squares   F-statistic:                     1.
Date:                Wed, 28 May 2025   Prob (F-statistic):              0.
Time:                        14:17:28   Log-Likelihood:                -176
No. Observations:                 305   AIC:                             35
Df Residuals:                     294   BIC:                             35
Df Model:                          10
Covariance Type:            nonrobust
==============================================================================
                                  coef    std err          t      P>|t|
------------------------------------------------------------------------------
const                         151.9600     16.230      9.363      0.000
LDA_Celebration/Desire        -41.6991     20.164     -2.068      0.040
LDA_Dreamy/Abstract Imagery    -5.1908     27.748     -0.187      0.852
LDA_Emotional/Inspirational   -38.3350     25.981     -1.476      0.141
LDA_Energetic/Upbeat Chant    -45.9124     24.021     -1.911      0.057
LDA_Fun/Party Anthem          -19.4600     21.127     -0.921      0.358
LDA_Journey/Escape            -83.0850     25.981     -3.198      0.002
LDA_Love & Friendship         -29.9600     24.021     -1.247      0.213
LDA_Relationship Conflict     -38.0538     21.661     -1.757      0.080
LDA_Romantic/Intimate         -36.7042     20.410     -1.798      0.073
LDA_Vulnerability/Overcoming  -52.5433     21.127     -2.487      0.013
==============================================================================
Omnibus:                       33.533   Durbin-Watson:                   1.
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               41.
Skew:                           0.857   Prob(JB):                     1.14e
Kurtosis:                       3.551   Cond. No.                          1
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is corr
```

```
# Prepare data: drop missing values
merged_df_clean = merged_df.dropna(subset=["Final_Points"])

# One-hot encode BERT topics
X = pd.get_dummies(merged_df_clean["BERT_Topic_Label"], prefix="BERT")
X = sm.add_constant(X)              # Add intercept term
X = X.astype(float)                 # Ensure numeric dtype

# Target variable
y = pd.to_numeric(merged_df_clean["Final_Points"], errors="coerce")

# Fit model
```

```
model = sm.OLS(y, X).fit()

# Show full statistical summary
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           Final_Points   R-squared:                       0.
Model:                            OLS   Adj. R-squared:                  0.
Method:                 Least Squares   F-statistic:                     1.
Date:                Wed, 28 May 2025   Prob (F-statistic):              0.
Time:                        14:17:28   Log-Likelihood:                -148
No. Observations:                 232   AIC:                             29
Df Residuals:                     221   BIC:                             30
Df Model:                          10
Covariance Type:            nonrobust
==============================================================================
                                  coef    std err          t      P>|t|
------------------------------------------------------------------------------
const                         225.7778     34.778      6.492      0.000
BERT_Chanting / Hype          -99.0159     47.394     -2.089      0.038
BERT_Dance & Party            -52.8578     45.611     -1.159      0.248
BERT_Emotional Breakup        -80.6667     44.898     -1.797      0.074
BERT_Feminine Power / Diva     39.8222     51.584      0.772      0.441
BERT_Intensity / Passion      -49.8403     50.697     -0.983      0.327
BERT_Nonsense / Playful Chant  -2.7190     49.901     -0.054      0.957
BERT_Performance & Confidence -44.2778     49.183     -0.900      0.369
BERT_Romantic Loneliness      -57.0278     44.576     -1.279      0.202
BERT_Seduction & Regret       -37.6278     47.938     -0.785      0.433
BERT_Self-Love / Creativity   -63.2222     44.898     -1.408      0.160
==============================================================================
Omnibus:                       45.427   Durbin-Watson:                   1.
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               66.
Skew:                           1.168   Prob(JB):                     2.87e
Kurtosis:                       4.212   Cond. No.                        1
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is corr
```

```
# Prepare data: drop missing values
merged_df_clean = merged_df.dropna(subset=["Final_Place"])

# One-hot encode BERT topics
X = pd.get_dummies(merged_df_clean["BERT_Topic_Label"], prefix="BERT")
X = sm.add_constant(X)              # Add intercept term
X = X.astype(float)                 # Ensure numeric dtype

# Target variable
y = pd.to_numeric(merged_df_clean["Final_Place"], errors="coerce")
```

```
# Fit model
model = sm.OLS(y, X).fit()

# Show full statistical summary
print(model.summary())
```

```
                                 OLS Regression Results
==============================================================================
Dep. Variable:            Final_Place   R-squared:                       0.
Model:                            OLS   Adj. R-squared:                  0.
Method:                 Least Squares   F-statistic:                     1.
Date:                Wed, 28 May 2025   Prob (F-statistic):              0.
Time:                        14:17:28   Log-Likelihood:                -787
No. Observations:                 232   AIC:                             15
Df Residuals:                     221   BIC:                             16
Df Model:                          10
Covariance Type:            nonrobust
==============================================================================
                                    coef    std err          t      P>|t|
------------------------------------------------------------------------------
const                            10.7222      1.745      6.146      0.000
BERT_Chanting / Hype              5.3254      2.378      2.240      0.026
BERT_Dance & Party                1.8778      2.288      0.821      0.413
BERT_Emotional Breakup            4.6852      2.252      2.080      0.039
BERT_Feminine Power / Diva       -1.2556      2.588     -0.485      0.628
BERT_Intensity / Passion          3.4653      2.543      1.363      0.174
BERT_Nonsense / Playful Chant     2.1601      2.503      0.863      0.389
BERT_Performance & Confidence     1.7222      2.467      0.698      0.486
BERT_Romantic Loneliness          3.3492      2.236      1.498      0.136
BERT_Seduction & Regret           1.5778      2.405      0.656      0.512
BERT_Self-Love / Creativity       3.8704      2.252      1.718      0.087
==============================================================================
Omnibus:                       77.609   Durbin-Watson:                   1.
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               12.
Skew:                          -0.013   Prob(JB):                       0.00
Kurtosis:                       1.869   Cond. No.                          1
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is corr
```

```
# Prepare data: drop missing values
merged_df_clean = merged_df.dropna(subset=["Semi_Place"])

# One-hot encode BERT topics
X = pd.get_dummies(merged_df_clean["BERT_Topic_Label"], prefix="BERT")
X = sm.add_constant(X)              # Add intercept term
X = X.astype(float)                 # Ensure numeric dtype

# Target variable
y = pd.to_numeric(merged_df_clean["Semi_Place"], errors="coerce")
```

```
# Fit model
model = sm.OLS(y, X).fit()

# Show full statistical summary
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             Semi_Place   R-squared:                       0.
Model:                            OLS   Adj. R-squared:                  0.
Method:                 Least Squares   F-statistic:                     1.
Date:                Wed, 28 May 2025   Prob (F-statistic):              0.
Time:                        14:17:28   Log-Likelihood:                -915
No. Observations:                 305   AIC:                             18
Df Residuals:                     294   BIC:                             18
Df Model:                          10
Covariance Type:            nonrobust
==============================================================================
                                   coef    std err          t      P>|t|
------------------------------------------------------------------------------
const                            7.6000      0.991      7.666      0.000
BERT_Chanting / Hype             0.7158      1.509      0.474      0.636
BERT_Dance & Party               0.4833      1.417      0.341      0.733
BERT_Emotional Breakup           2.5471      1.306      1.950      0.052
BERT_Feminine Power / Diva       0.7750      1.587      0.488      0.626
BERT_Intensity / Passion         2.0400      1.402      1.455      0.147
BERT_Nonsense / Playful Chant    0.1000      1.487      0.067      0.946
BERT_Performance & Confidence    2.7333      1.270      2.152      0.032
BERT_Romantic Loneliness         1.7000      1.264      1.345      0.180
BERT_Seduction & Regret          1.7226      1.332      1.293      0.197
BERT_Self-Love / Creativity      0.8063      1.323      0.609      0.543
==============================================================================
Omnibus:                       80.228   Durbin-Watson:                   2.
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               14.
Skew:                           0.016   Prob(JB):                     0.000
Kurtosis:                       1.929   Cond. No.                        1
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is corr
```

```
# Prepare data: drop missing values
merged_df_clean = merged_df.dropna(subset=["Semi_Points"])

# One-hot encode BERT topics
X = pd.get_dummies(merged_df_clean["BERT_Topic_Label"], prefix="BERT")
X = sm.add_constant(X)              # Add intercept term
X = X.astype(float)                 # Ensure numeric dtype

# Target variable
```

```
y = pd.to_numeric(merged_df_clean["Semi_Points"], errors="coerce")

# Fit model
model = sm.OLS(y, X).fit()

# Show full statistical summary
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            Semi_Points   R-squared:                       0.
Model:                            OLS   Adj. R-squared:                  0.
Method:                 Least Squares   F-statistic:                     1.
Date:                Wed, 28 May 2025   Prob (F-statistic):             0.0
Time:                        14:17:28   Log-Likelihood:                -176
No. Observations:                 305   AIC:                             35
Df Residuals:                     294   BIC:                             35
Df Model:                          10
Covariance Type:            nonrobust
==============================================================================
                                    coef    std err          t      P>|t|
------------------------------------------------------------------------------
const                           151.9600     16.216      9.371      0.000
BERT_Chanting / Hype            -56.8547     24.677     -2.304      0.022
BERT_Dance & Party              -49.3350     23.170     -2.129      0.034
BERT_Emotional Breakup          -55.4012     21.361     -2.594      0.010
BERT_Feminine Power / Diva      -50.3350     25.958     -1.939      0.053
BERT_Intensity / Passion        -17.5200     22.932     -0.764      0.445
BERT_Nonsense / Playful Chant   -22.6100     24.323     -0.930      0.353
BERT_Performance & Confidence   -51.9344     20.773     -2.500      0.013
BERT_Romantic Loneliness        -44.3100     20.671     -2.144      0.033
BERT_Seduction & Regret         -30.2503     21.795     -1.388      0.166
BERT_Self-Love / Creativity     -10.3350     21.642     -0.478      0.633
==============================================================================
Omnibus:                       29.811   Durbin-Watson:                   1.
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               35.
Skew:                           0.813   Prob(JB):                     1.87e
Kurtosis:                       3.393   Cond. No.                         1
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is corr
```

```
import statsmodels.api as sm
import pandas as pd

# Clean 'Sex' column
sex_map = {"F": 0, "M": 1, "Mixed": 2}
merged_df["Sex_Clean"] = merged_df["Sex"].map(sex_map)

# Convert 'National_Language_Used' column to int (True/False to 1/0)
```

```
merged_df["National_Language_Used"] = merged_df["National_Language_Used"].astyp

# Replace the original column in features list with cleaned version
features_to_test = [
    "Solo_Artist", "Returning_Artist_Ind", "Number of Members",
    "Multiple_Language", "EU", "National_Language_Used", "Sex_Clean"
]

# Drop missing target values
df_clean = merged_df.dropna(subset=["Grand_Final_Ind"])
y = df_clean["Grand_Final_Ind"].astype(int)

for feature in features_to_test:
    X = df_clean[[feature]].copy()

    # Handle categorical variables
    if X[feature].dtype == "object":
        X = pd.get_dummies(X, drop_first=True)

    # Ensure all values are float for statsmodels
    X = X.apply(pd.to_numeric, errors='coerce').fillna(0)
    X = sm.add_constant(X)

    try:
        model = sm.Logit(y, X).fit(disp=False)
        print(f"\n=== Logistic Regression for: {feature} ===")
        print(model.summary())
    except Exception as e:
        print(f"\n[Error fitting model for: {feature}] {e}")
```

```
converged:                       True   LL-Null:                       -231
Covariance Type:            nonrobust   LLR p-value:                   0.09
=============================================================================
                      coef    std err          z      P>|z|      [0.025
-----------------------------------------------------------------------------
const               0.5367      0.122      4.410      0.000       0.298
Multiple_Language   0.4850      0.301      1.613      0.107      -0.104
=============================================================================

=== Logistic Regression for: EU ===
                           Logit Regression Results
=============================================================================
Dep. Variable:         Grand_Final_Ind   No. Observations:
Model:                           Logit   Df Residuals:
Method:                            MLE   Df Model:
Date:                 Wed, 28 May 2025   Pseudo R-squ.:                 0.003
Time:                         14:17:28   Log-Likelihood:                -230
converged:                        True   LL-Null:                       -231
Covariance Type:             nonrobust   LLR p-value:                   0.1
-----------------------------------------------------------------------------
```

```
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.9
------------------------------------------------------------------------------
const          0.4520      0.171      2.644      0.008       0.117       0.
EU             0.2906      0.225      1.292      0.196      -0.150       0.
==============================================================================

=== Logistic Regression for: National_Language_Used ===
                           Logit Regression Results
==============================================================================
Dep. Variable:        Grand_Final_Ind   No. Observations:
Model:                          Logit   Df Residuals:
Method:                           MLE   Df Model:
Date:                Wed, 28 May 2025   Pseudo R-squ.:                   0.03
Time:                        14:17:28   Log-Likelihood:                  -223
converged:                       True   LL-Null:                         -231
Covariance Type:            nonrobust   LLR p-value:                   4.509e
==============================================================================
                             coef    std err          z      P>|z|      [0.
------------------------------------------------------------------------------
const                      0.2955      0.135      2.192      0.028       0.
National_Language_Used     0.9816      0.250      3.934      0.000       0.
==============================================================================

=== Logistic Regression for: Sex_Clean ===
                           Logit Regression Results
==============================================================================
Dep. Variable:        Grand_Final_Ind   No. Observations:
Model:                          Logit   Df Residuals:
Method:                           MLE   Df Model:
Date:                Wed, 28 May 2025   Pseudo R-squ.:                 7.699e
Time:                        14:17:28   Log-Likelihood:                  -231
converged:                       True   LL-Null:                         -231
Covariance Type:            nonrobust   LLR p-value:                      0.8
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.9
------------------------------------------------------------------------------
const          0.6438      0.157      4.090      0.000       0.335       0.
Sex_Clean     -0.0309      0.164     -0.189      0.850      -0.351       0.
==============================================================================
```

Model Evaluation

```
# One-hot encode topics and categorical variables
topic_dummies = pd.get_dummies(merged_df["BERT_Topic_Label"], prefix="Topic")
country_dummies = pd.get_dummies(merged_df["Country_Merge"], prefix="Country")
group_dummies = pd.get_dummies(merged_df["Country_Group"], prefix="Group")
```

```
# Define target variable for classification: whether the song reached the Granc
y_class = merged_df["Grand_Final_Ind"]


# Use only topic dummy variables as the feature set for modeling
X_topics = topic_dummies


 # Combine topic and country dummy variables as features
X_topic_country = pd.concat([topic_dummies, country_dummies], axis=1)



X_topic_country_artist = pd.concat([
    topic_dummies, # Topic-related features (e.g., LDA or BERT topics)
    country_dummies, # Country dummy variables
    group_dummies, # Regional groupings of countries
    merged_df[["Solo_Artist", "Returning_Artist_Ind", "Number of Members"]]
], axis=1).astype(float)



X_topic_country_artist_language = pd.concat([
    topic_dummies,
    country_dummies,
    group_dummies,
    merged_df[["Solo_Artist", "Returning_Artist_Ind", "Multiple_Language", "Num
], axis=1).astype(float)



X_long = pd.concat([
    topic_dummies,
    country_dummies,
    group_dummies,
    merged_df[["Solo_Artist", "Returning_Artist_Ind", "Multiple_Language", "Num
], axis=1).astype(float)
```

```
# Define numeric features
numeric_features = merged_df[[
    "Solo_Artist", "Returning_Artist_Ind", "Number of Members",
    "Multiple_Language", "EU", "National_Language_Used", "Sex_Clean"
]]

# Combine all features into design matrix X
X_full = pd.concat([
    topic_dummies,
    country_dummies,
    group_dummies,
    numeric_features,
], axis=1).astype(float)


from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score, f1_score


for name, X_variant in [("Topics Only", X_topics), ("Topic+Country", X_topic_co
    X_train, X_test, y_train, y_test = train_test_split(X_variant, y_class, tes

    clf = RandomForestClassifier(random_state=42, class_weight="balanced")
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)

    print(f"\n=== Classification Report: {name} ===")
    print(classification_report(y_test, y_pred))
```

```
    accuracy                           0.42        72
   macro avg       0.45      0.45      0.42        72
weighted avg       0.49      0.42      0.42        72


=== Classification Report: Topic+Country ===
              precision    recall  f1-score   support

           0       0.53      0.40      0.45        25
           1       0.72      0.81      0.76        47

    accuracy                           0.67        72
   macro avg       0.62      0.60      0.61        72
weighted avg       0.65      0.67      0.65        72


=== Classification Report: Topic+Country+Artist ===
              precision    recall  f1-score   support

           0       0.53      0.40      0.45        25
           1       0.72      0.81      0.76        47
```

```
           1       0.72      0.81      0.76        47

    accuracy                           0.67        72
   macro avg       0.62      0.60      0.61        72
weighted avg       0.65      0.67      0.65        72


=== Classification Report: Topic+Country+Artist+Language ===
              precision    recall  f1-score   support

           0       0.61      0.44      0.51        25
           1       0.74      0.85      0.79        47

    accuracy                           0.71        72
   macro avg       0.68      0.65      0.65        72
weighted avg       0.70      0.71      0.69        72


=== Classification Report: Longlist ===
              precision    recall  f1-score   support

           0       0.62      0.40      0.49        25
           1       0.73      0.87      0.80        47

    accuracy                           0.71        72
   macro avg       0.68      0.64      0.64        72
weighted avg       0.69      0.71      0.69        72


=== Classification Report: full ===
              precision    recall  f1-score   support

           0       0.64      0.36      0.46        25
           1       0.72      0.89      0.80        47

    accuracy                           0.71        72
   macro avg       0.68      0.63      0.63        72
weighted avg       0.70      0.71      0.68        72
```