Pop Song Analysis
PROJECT

Text Technology

Structuring and Analyzing
Pop-Song Data

Nayoung Kim

[1] Project Topic – Original Plan

Original Project Topic :

Trend Analysis of Recent Pop Song Lyrics Using the Billboard Hot-100 Songs

[1] Project Topic – Original Plan

Trend Analysis
of Recent Pop Song Lyrics
Using the Billboard Hot-100 Songs

<Original Plan>

Topic

"Lyric" Analysis of Billboard Hot 100 Songs Reflecting Current Pop Trends

Reason

Because lyrics are publicly available data, they would be useful for large-scale collection

I already have a good understanding of this domain,
because listening to music is one of my hobbies

Detailed Plan

1. Collect data through crawling/scraping and APIs
2. Process the data using Python
3. Save it as a JSON file and structure it with an XSD file
4. Store it in a DBMS for analysis and easy access

[1] Project Topic – Original Plan

Trend Analysis
of Recent Pop Song Lyrics
Using the Billboard Hot-100 Songs

<Problem>

- **Song information was easily obtained through APIs**
  However, not all APIs provide lyrics

- **Large-scale data crawling often results in frequent site blocks,**
  and bypassing them may lead to copyright issues

Project halted due to difficulties in obtaining "lyrics(key data)"

- Link to the Python script(Github) :
https://github.com/witerk/Stuttgart_TextTechnology/blob/326addc48ee10820cd88166f86a2ebdac8d41619/Collect_Data-Lyrics.ipynb

[1] Project Topic – New Plan

Focused mainly on lyric data

~~Trend Analysis of Recent Pop Song Lyrics~~

Change Topic

Structuring and Analyzing Pop Song

Need more diverse data

Re-collect to update data

[2] Data Collection

<Collect Data>
: Using "Spotify" Songs data

- Kaggle : https://www.kaggle.com/datasets/joebeachcapital/30000-spotify-songs?resource=download

- Almost 30,000 Songs Data from the Spotify API
- Proivde 23 classes

Select
12 classes

track_id
track_name
track_artist
track_popularity
track_album_id
track_album_name
track_album_release_date
playlist_name
playlist_id
playlist_genre
playlist_subgenre

danceability
energy
key
loudness
mode
speechiness
acousticness
instrumentalness
liveness
valence
tempo
duration_ms

Structuring and Analyzing Pop Song

<1. Renaming Columns>

- track_id → Song_ID
- track_name → Song_NAME
- track_artist → Song_ARTIST
- track_popularity → Song_POPULARITY
- track_album_id → Album_ID
- track_album_name → Album_NAME
- track_album_release_date → Album_DATE
- danceability → Style_DANCE
- energy → Style_ENERGY
- valence → Song_VALENCE
- tempo → Style_TEMPO
- duration_ms → Song_PLAYTIME

<2. Add New Columns>

xxxx-xx-xx format

Album_DATE

- Plan to retrieve data at least by quarter

Album_DATE_YEAR

Album year (YYYY)

Album_DATE_QUARTER

Quarter 1 to Quarter 4 (q)

- If only year remains in some dates
→ mark as 'No data'

[3] Data Preparation

Structuring and Analyzing Pop Song

<2. Add New Columns>

Style_DANCE

(Range 0-1)
label as 'Danceable' if ≥ 0.7

Style_ENERGY

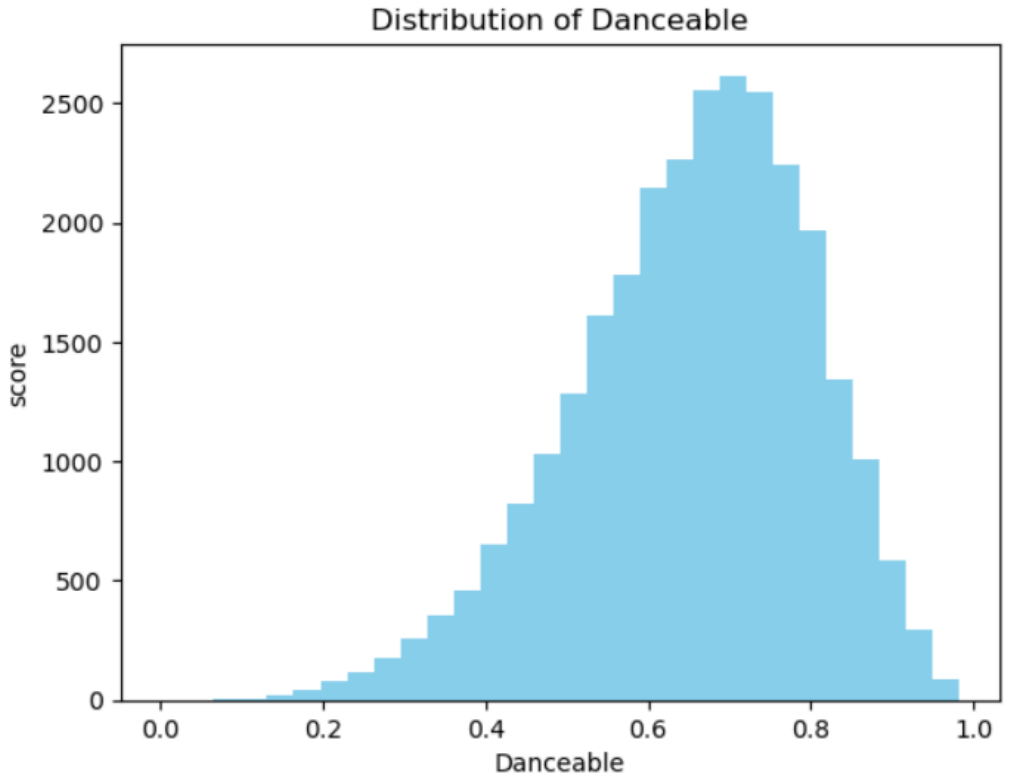
(Range 0-1)
label as 'Relaxed' if ≤ 0.4
label as 'Energetic' if ≥ 0.7

Style_TEMPO

label as 'Slow-paced' if ≤ 90
label as 'Fast-paced' if ≥ 140

↓

Song_STYLE



*Feature thresholds set with reference to graphs and domain insights

[3] Data Preparation

Structuring and Analyzing Pop Song

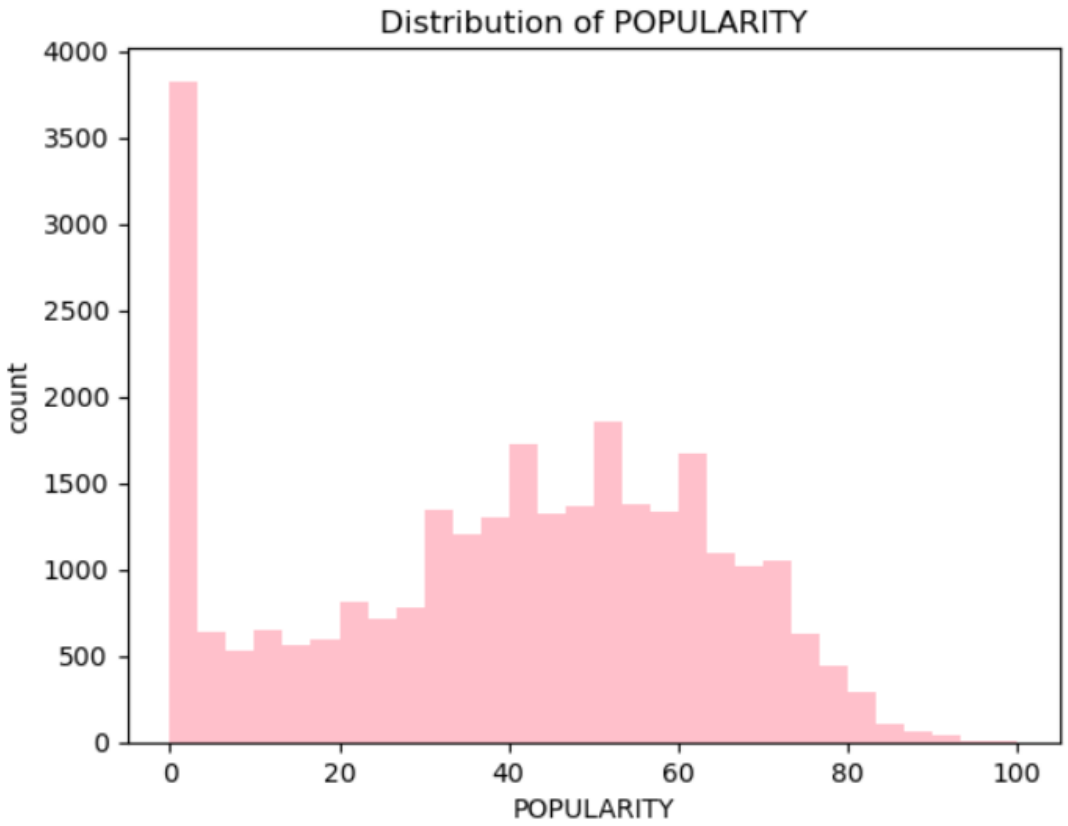
<2. Add New Columns>

Song_ POPULARITY (Range 0-100)

↓

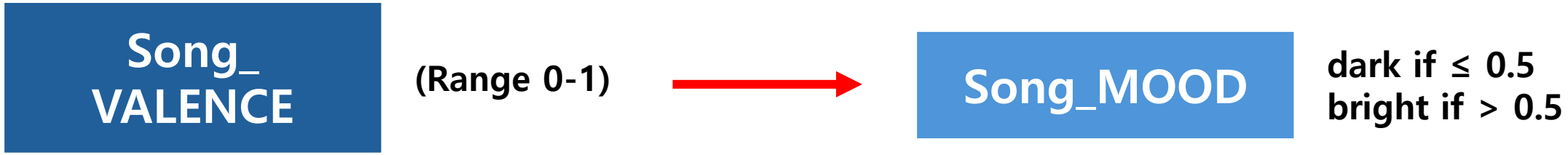
POPULARITY _RANK

★☆☆☆☆ if (< 30)
★★☆☆☆ if (≥ 30) & (< 50)
★★★☆☆ if (≥ 50) & (< 70)
★★★★☆ if (≥ 70) & (< 80)
★★★★★ if (≥ 80)



*Feature thresholds set with reference to graphs and domain insights

<2. Add New Columns>



<3. Data cleaning>

Song_ PLAYTIME

- Change unit from milliseconds(ms) to seconds(s)

Style_DANCE

Song_ VALENCE

Style_ENERGY

Style_TEMPO

- Very small decimal values
  → displayed as non-numeric (scientific notation)
  → converted to 0

<3. Data cleaning>

Song_NAME

Album_NAME

Song_ARTIST

- Missing values filled with empty strings
- special characters removed

<4. Save to JSON>


```
[{'Album_DATE': '2019-06-14',
  'Album_DATE_QUARTER': '2',
  'Album_DATE_YEAR': '2019',
  'Album_ID': '2oCs0DGTsRO98Gh5ZSl2Cx',
  'Album_NAME': 'I Don&#x27;t Care (with Justin Bieber) [Loud Luxury Remix]',
  'POPULARITY_RANK': '★★★☆☆',
  'Song_ARTIST': 'Ed Sheeran',
  'Song_ID': '6f807x0ima9a1j3VPbc7VN',
  'Song_MOOD': 'bright',
  'Song_NAME': 'I Don&#x27;t Care (with Justin Bieber) - Loud Luxury Remix',
  'Song_PLAYTIME': 194,
  'Song_POPULARITY': 66,
  'Song_STYLE': 'Energetic ',
  'Song_VALENCE': 0.518,
  'Style_DANCE': 0.748,
  'Style_ENERGY': 0.916,
  'Style_TEMPO': 122.036},
```


*pprint(songs_json_df)

[4] Structured in XML

Structuring and Analyzing Pop Song

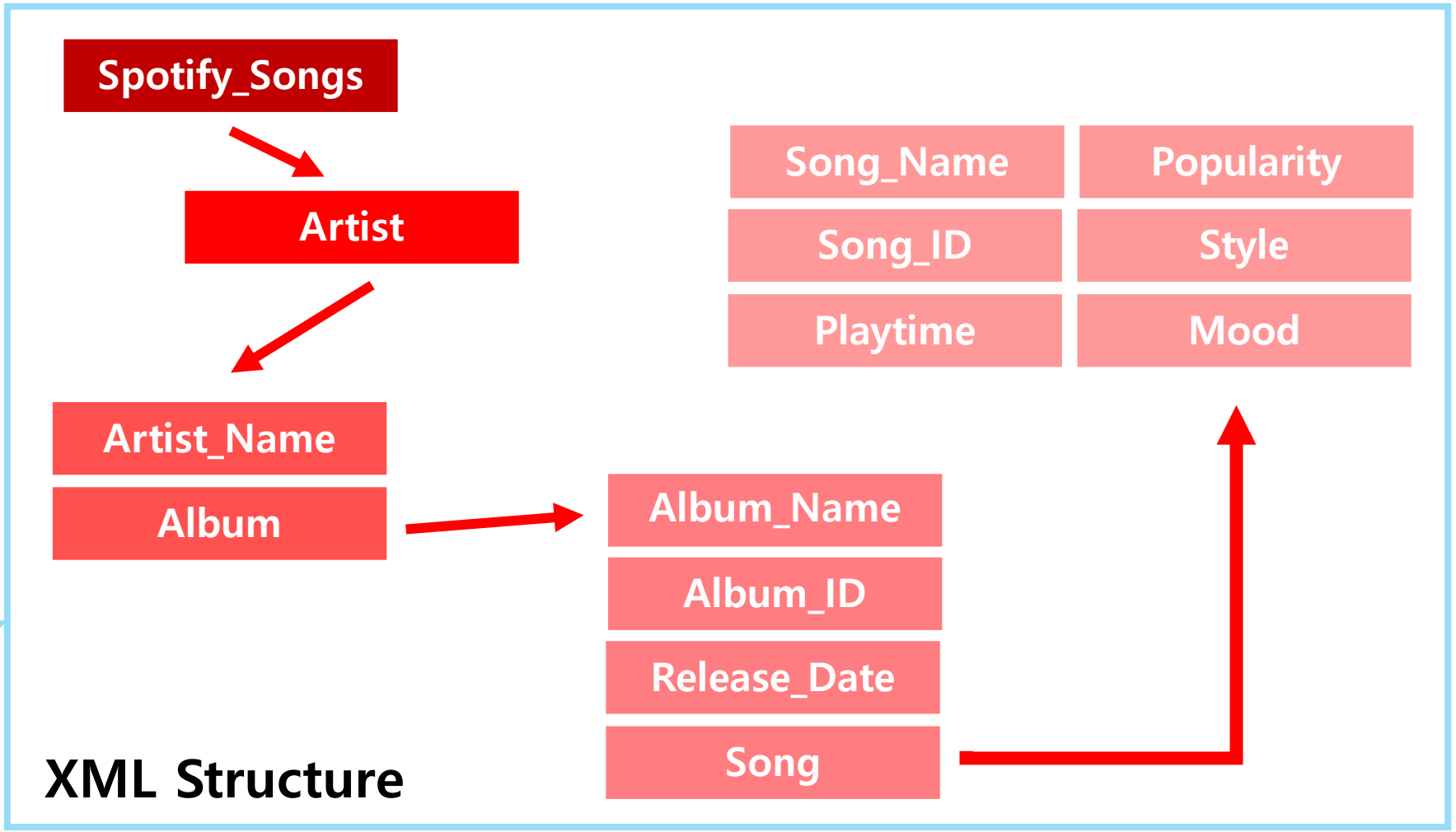
<1. Generate XML>

Flattened data

JSON

Hierarchically structured data

XML



[4] Structured in XML

```xml
<?xml version="1.0" encoding="UTF-8"?>

<Spotify_Songs xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:noNamespaceSchemaLocation="spotify_schema.xsd">

 <Artist>
  <Artist_Name>!!!</Artist_Name>
  <Album>
   <Album_Name>Thr!!!er</Album_Name>
   <Album_ID>3ywERU9rVKEpCQE50fQOgM</Album_ID>
   <Release_Date Year="2013" Quarter="No data">2013</Release_Date>
   <Song>
    <Song_Name>One Girl / One Boy</Song_Name>
    <Song_ID>7y8aVfDkqt6qirGNivvs0M</Song_ID>
    <Playtime Duration="Second">243</Playtime>
    <Popularity rank="★★☆☆☆">[48/100]</Popularity>
    <Style Dance="0.702" Energy="0.851" Tempo="117.004">Energetic </Style>
    <Mood Valence="0.87">bright</Mood>
   </Song>
   <Song>
    <Song_Name>Even When The Water's Cold</Song_Name>
    <Song_ID>2UjEyfnKzaY7qpBEeESJjv</Song_ID>
    <Playtime Duration="Second">227</Playtime>
    <Popularity rank="★★★☆☆">[52/100]</Popularity>
    <Style Dance="0.709" Energy="0.831" Tempo="104.971">Energetic </Style>
    <Mood Valence="0.866">bright</Mood>
   </Song>
  </Album>
 </Artist>
```

Structuring and Analyzing Pop Song

<1. Generate XML>

Structured XML was generated from bulk JSON data using Python (*for details, refer to the code file)

Check Well-formedness & Validate

- Spotify Data – Python Code(Github) : https://github.com/witerk/Stuttgart_TextTechnology/blob/d2b44311f484ee448181b4bdafab677d86042453/Collect_Data-Spotify.ipynb

XML

[4] Structured in XML

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

 <xs:element name="Spotify_Songs">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="Artist" maxOccurs="unbounded">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="Artist_Name" type="xs:string"/>
       <xs:element name="Album" maxOccurs="unbounded">
        <xs:complexType>
         <xs:sequence>
          <xs:element name="Album_Name" type="xs:string"/>
          <xs:element name="Album_ID" type="xs:string"/>
          <xs:element name="Release_Date">
           <xs:complexType>
            <xs:simpleContent>
             <xs:extension base="xs:string">
              <xs:attribute name="Year" type="xs:gYear"/>
              <xs:attribute name="Quarter" type="xs:string"/>
             </xs:extension>
            </xs:simpleContent>
           </xs:complexType>
          </xs:element>
          <xs:element name="Song" maxOccurs="unbounded">
           <xs:complexType>
            <xs:sequence>
             <xs:element name="Song_Name" type="xs:string"/>
             <xs:element name="Song_ID" type="xs:string"/>
```

Structuring and Analyzing Pop Song

<2. Generate XSD>

Existing XML structure used to generate XSD schema

Check Well-formedness & Validate

- Spotify XML & XSD (Github) : https://github.com/witerk/Stuttgart_TextTechnology/blob/d2b44311f484ee448181b4bdafab677d86042453/Collect_Data-Spotify.ipynb

XSD

[5] Store Data in a Database

<Create table>

- Finalized data stored in the database
- Split into three tables to remove duplicates and improve consistency

ARTIST	ALBUM	SONGS
ARTIST_ID	ALBUM_ID	ALBUM_ID
ARTIST_NAME	ALBUM_NAME	SONG_ID
	RELEASE_YEAR	SONG_NAME
	RELEASE_QUARTER	SONG_POPULARITY
		DANCEABLE
		ENGERGETIC
		TEMPO
		MOOD
		PLAYTIME

[5] Store Data in a Database

<Create table>

- Finalized data stored in the database
- Split into three tables to remove duplicates and improve consistency

ARTIST

ARTIST_ID	ARTIST _NAME
PRIMARY KEY NUT NULL NUMBER LENGTH 22	VARCHAR2 LENGTH 126

ALBUM

SONGS

[5] Store Data in a Database

<Create table>

- Finalized data stored in the database
- Split into three tables to remove duplicates and improve consistency

ARTIST | **ALBUM** | **SONGS**

ALBUM_ID	ALBUM_NAME	ARTIST_ID	RELEASE_YEAR	RELEASE_QUARTER
PRIMARY KEY NUT NULL VARCHAR2 LENGTH 126	VARCHAR2 LENGTH 128	VARCHAR2 LENGTH 128	NUMBER LENGTH 22	NUMBER LENGTH 22

[5] Store Data in a Database

Structuring and Analyzing Pop Song

<Create table>

- Finalized data stored in the database
- Split into three tables to remove duplicates and improve consistency

ARTIST | **ALBUM** | **SONGS**

ALBUM_ID	SONG_ID	SONG _NAME	SONG_ POPULARITY	DANCEABLE	ENGERGETIC	TEMPO	MOOD	PLAYTIME
VARCHAR2 LENGTH 126	PRIMARY KEY NUT NULL VARCHAR2 LENGTH 126	VARCHAR2 LENGTH 128	NUMBER LENGTH 22	NUMBER LENGTH 22	NUMBER LENGTH 22	NUMBER LENGTH 22	NUMBER LENGTH 22	NUMBER LENGTH 22

[5] Store Data in a Database

Structuring and Analyzing Pop Song

<Create table>

[5] Store Data in a Database

Structuring and Analyzing Pop Song

<Create table>

DELETE

ALBUM _DATE	POLULARITY _RANK	SONG _STYLE	SONG _MOOD

Columns designed for readability and analysis
Not included in storage-oriented DBMS
“Original data retained, duplicates removed”

Structuring and Analyzing Pop Song

1. Characteristics of Popular Songs

Quarterly Top 100 Songs by Year

- **In some quarters (e.g., early 1900s), fewer than 100 songs were available**
- **Songs with 0 popularity were also included in the selection**

→ Top 10% Songs by Year

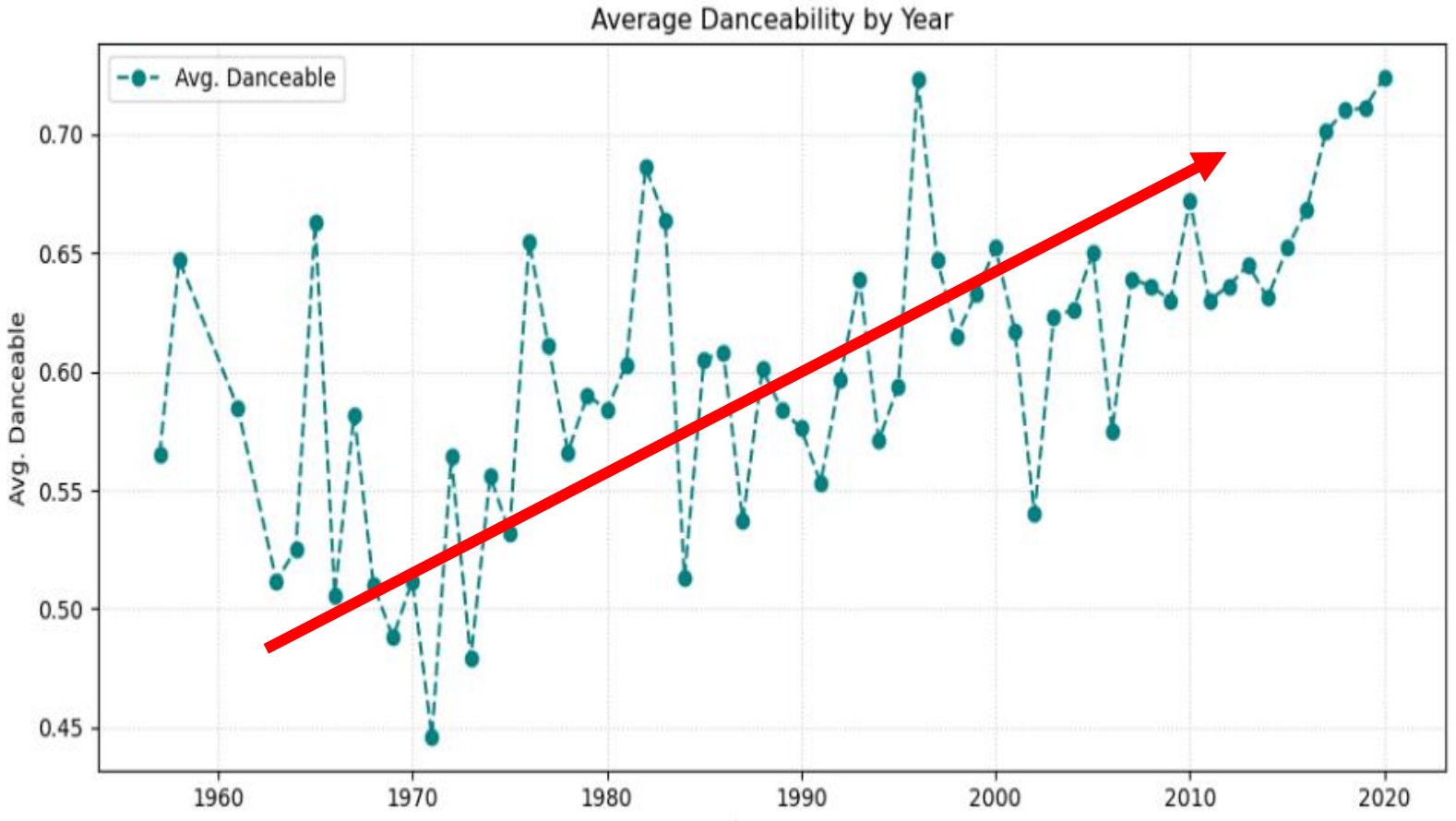
(*Popularity threshold was not applied, as older songs tend to have lower overall popularity)

- **Analyzed averages of danceability, energy, tempo, and mood**

	RELEASE_YEAR	AVG_DANCEABLE	AVG_ENERGETIC	AVG_TEMPO	AVG_MOOD
1	1957	0.565	0.962	148.8	0.906
2	1958	0.647	0.582	167.4	0.915
3	1961	0.585	0.036	151.2	0.433
4	1963	0.512	0.513	148.7	0.688

Structuring and Analyzing Pop Song

1. Characteristics of Popular Songs

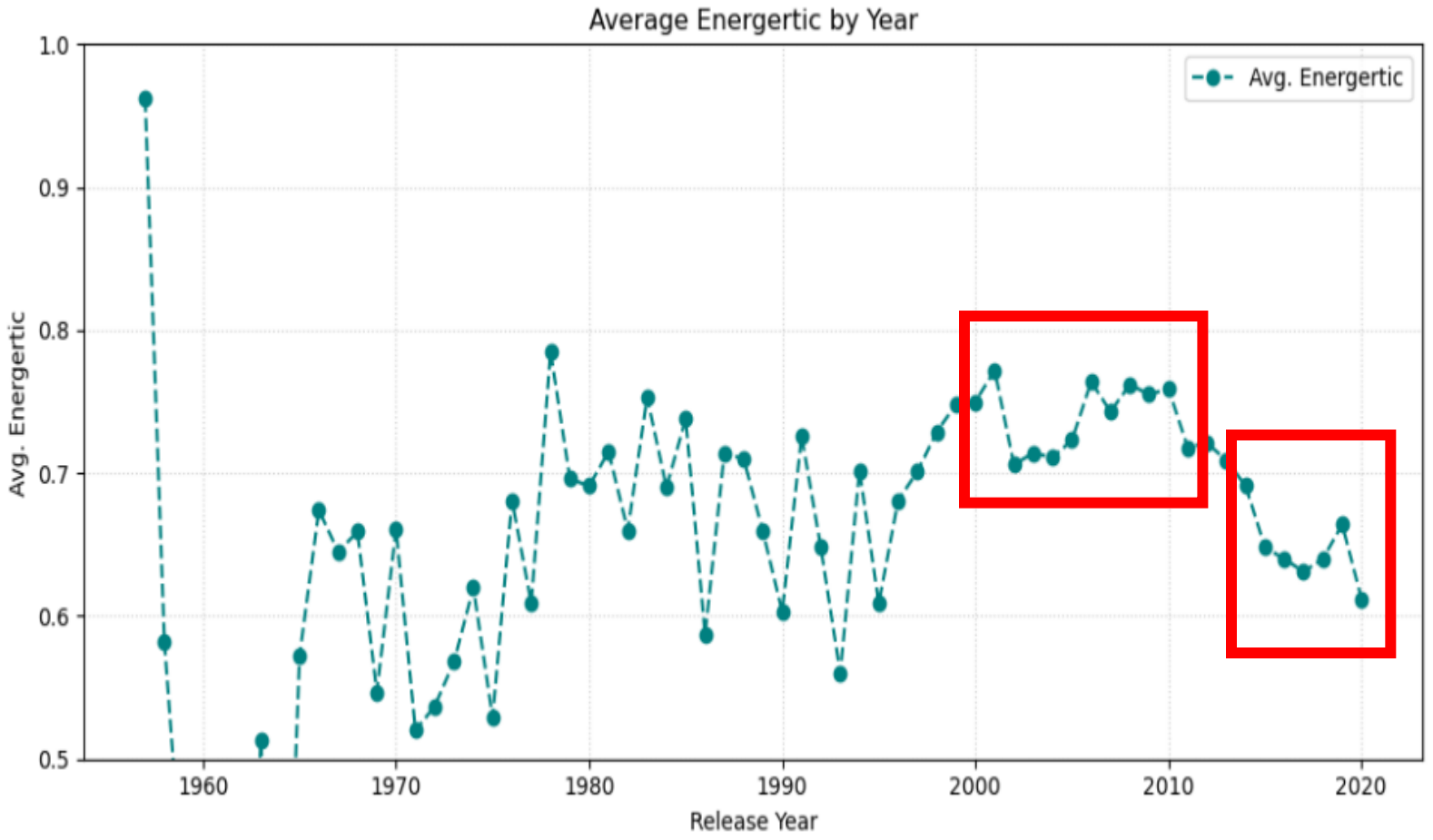


- **Danceability**
  : Gradual improvement toward the modern era

(*Python Visualization)

Structuring and Analyzing Pop Song

1. Characteristics of Popular Songs



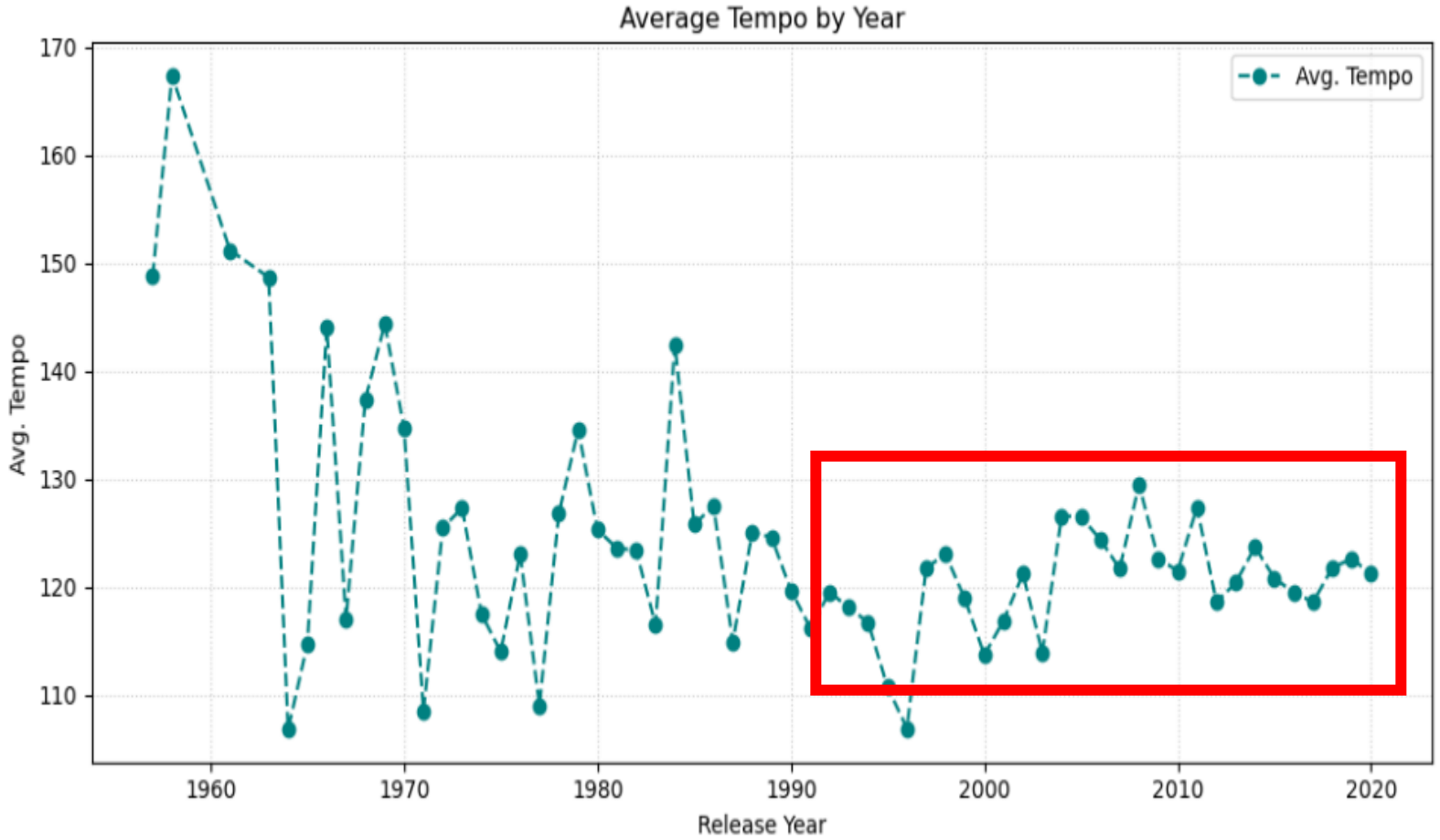
(*Python Visualization)

- **Energetic**
  : Energetic songs were popular in the 2000s

  However, energy levels declined again between 2010 and 2020

Structuring and Analyzing Pop Song

1. Characteristics of Popular Songs



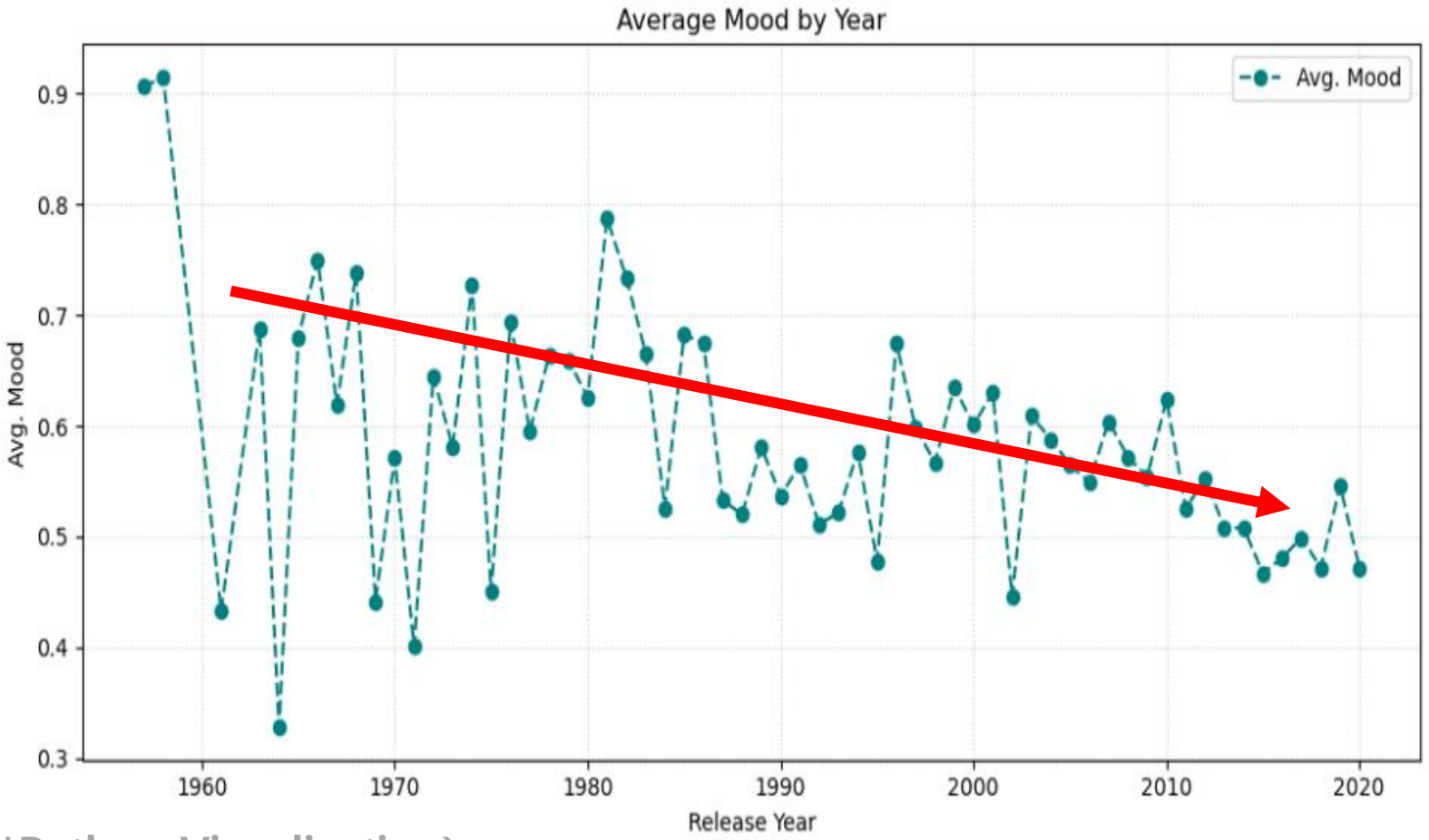
(*Python Visualization)

- **Tempo**
  : Overall downward trend

  Similar to the energy graph, there was an increase in the late 2010s,
  followed by a declining tendency between 2010 and 2020

Structuring and Analyzing Pop Song

1. Characteristics of Popular Songs



(*Python Visualization)

- **Mood**
  : Overall downward trend

  In the early 2000s, songs with a bright mood were more popular,
  but as we moved into the 2020s, the overall mood of songs became darker

2. Quarterly Analysis

It was assumed that seasonal changes (by quarter) might influence song characteristics
However, the analysis showed no significant differences

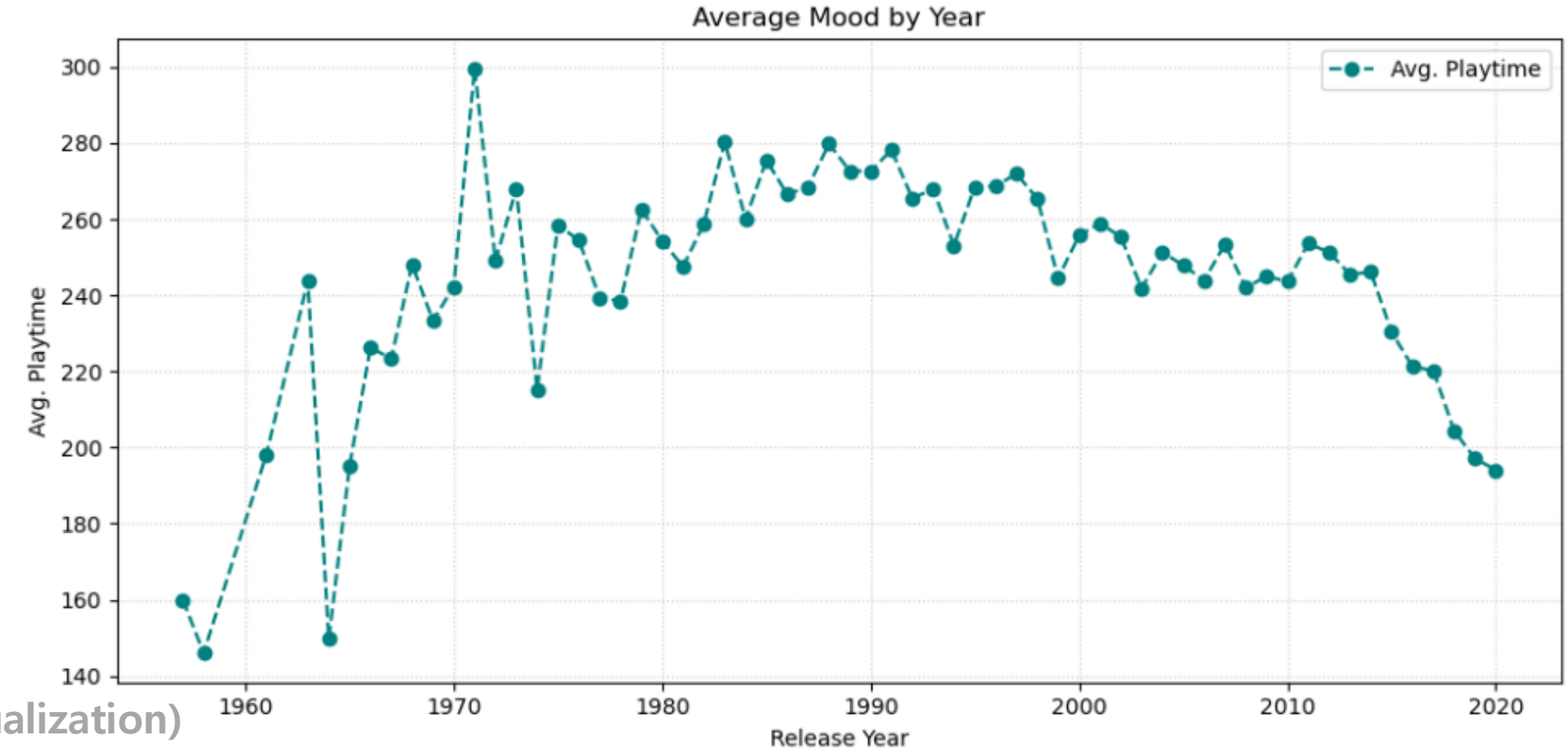
	RELEASE_QUARTER	AVG_POPULARITY	AVG_DANCEABLE	AVG_ENERGETIC	AVG_TEMPO	AVG_MOOD
1	1	36.8	0.649	0.702	120.5	0.528
2	2	39.1	0.655	0.704	121.2	0.509
3	3	39	0.658	0.699	120.7	0.503
4	4	40.5	0.659	0.693	121.1	0.496

The same result was observed even when analyzing only the top 10% most popular songs

	RELEASE_QUARTER	AVG_POPULARITY	AVG_DANCEABLE	AVG_ENERGETIC	AVG_TEMPO	AVG_MOOD
1	1	72	0.66	0.679	121.1	0.541
2	2	74.3	0.669	0.676	121.1	0.548
3	3	73.6	0.664	0.679	122.8	0.518
4	4	75.1	0.673	0.662	122	0.526

3. Playtime by Year

Since 2010, we can observe that the playtime of songs has decreased significantly-dropping by about one minute, from around 4 minutes to 3 minutes



(*Python Visualization)

[7] Challenges / Difficulties

Structuring and Analyzing Pop Song

1. Pop song - Lyrics Data Collection

- Official streaming service APIs
  → lyrics not provided

- Via Google search
  → blocked by anti-crawling/scraping mechanisms

- Direct access via 'Genius' URLs
  → fails with slight variation in artist/song names
  (e.g., "Charli xcx & Billie" vs "Charli xcx, Billie")

- Using 'googlesearch' to access top search result
  → possible by adding headers(User-Agent), rotating IPs, and randomizing request intervals
  → Data collection technically feasible but raises copyright issues
  → topic changed

2. Spotify Data - Cleaning

- Although the Spotify Songs dataset was officially distributed, it was not pre-cleaned
  → Removed duplicates
  Handled outliers
  Set default values for nulls
  Adjusted units and scaling
  Removed special characters
  Outliers were especially hard to detect, requiring careful manual review

[8] References

1) Billboard-Chat unofficial API,
Github: KoreanThinker/billboard-json,
https://github.com/KoreanThinker/billboard-json

2) Spotify DataSet,
Kaggle: 30000 Spotify Songs,
https://www.kaggle.com/datasets/joebeachcapital/30000-spotify-songs?resource=download