Spotify Song Recommendation System

Importing librarires

In [ ]:

```
import pandas as pd
```

Loading dataset

In [ ]:

```
df = pd.read_csv('spotify_millsongdata.csv')
df.head()
```

Out[ ]:

	artist	song	link	text
0	ABBA	Ahe's My Kind Of Girl	/a/abba/ahes+my+kind+of+girl_20598417.html	Look at her face, it's a wonderful face \r\nA...
1	ABBA	Andante, Andante	/a/abba/andante+andante_20002708.html	Take it easy with me, please \r\nTouch me gen...
2	ABBA	As Good As New	/a/abba/as+good+as+new_20003033.html	I'll never know why I had to go \r\nWhy I had...
3	ABBA	Bang	/a/abba/bang_20598415.html	Making somebody happy is a question of give an...
4	ABBA	Bang-A-Boomerang	/a/abba/bang+a+boomerang_20002668.html	Making somebody happy is a question of give an...

Data Exploration

In [ ]:

```
df.shape
```

Out[ ]:

```
(57650, 4)
```

In [ ]:

```
df.isnull().sum()
```

Out[ ]:

```
artist    0
song      0
link      0
text      0
dtype: int64
```

In [ ]:

```
df = df.sample(5000).drop('link', axis=1).reset_index(drop=True)
df.shape
```

Out[ ]:

```
(5000, 3)
```

In [ ]:

```
df.head()
```

Out[ ]:

	artist	song	text
0	INXS	Make Your Peace	There are rivers running \r\nJust for you and...
1	Conway Twitty	Happy Birthday Darlin'	Hello darlin' happy birthday \r\nI've decided...
2	Keith Urban	Til' Summer Comes Around	Another long summer's come and gone \r\nI don...
3	Ella Fitzgerald	A Fine Romance	A fine romance, with no kisses \r\nA fine rom...
4	Gordon Lightfoot	Looking At The Rain	Looking at the rain \r\nFeeling the pain \r\...

In [ ]:

```
df['text'][0]
```

Out[ ]:

```
"There are rivers running  \r\nJust for you and me  \r\nIn the darkest hours  \r\n
Choices made to be  \r\nI choose to pull my punches  \r\nDon't you test my conscie
nce  \r\nYou'll see  \r\nYou'll see  \r\nCut your teeth and make your peace  \r\nT
hat's what you asked for  \r\n  \r\nGot a way of getting  \r\nWhat I want to see
\r\nWhen you know I'm winning  \r\nYou'll get over me  \r\nAll the trash you're th
inking  \r\nDon't you feel it sinking  \r\nYou'll see  \r\nYou'll see  \r\n  \r\nA
ll this bitter fighting  \r\nMakes no sense to me  \r\nWe're only talking circles
\r\nWe're losing sympathy  \r\nNo time like the future  \r\nMake signs like I need
ya  \r\nYou see  \r\nYou see  \r\n  \r\nCut your teeth and make your peace  \r\nTh
at's what you ask for  \r\nCut your teeth and make your peace  \r\nIs what you ask
for\r\n\r\n"
```

Text Preprocessing

In [ ]:

```
df['text'] = df['text'].str.lower().replace(r'^\w\s', ' ').replace(r'\n', ' ', rege
df['text']
```

Out[ ]:

```
0       there are rivers running  \r just for you and ...
1       hello darlin' happy birthday  \r i've decided ...
2       another long summer's come and gone  \r i don'...
3       a fine romance, with no kisses  \r a fine roma...
4       looking at the rain  \r feeling the pain  \r o...
                              ...
4995    raymond versus raymond  \r   \r there's three ...
4996    will i live tomorrow?  \r well i just can't sa...
4997    every now and then we find a special friend  \...
4998    have you heard? the coast of maine just got ca...
4999    [verse 1]  \r i'm like a child looking off in ...
Name: text, Length: 5000, dtype: object
```

In [ ]:

```
df.tail()
```

Out[ ]:

	artist	song	text
4995	Usher	Monstar	raymond versus raymond \r \r there's three ...
4996	Jimi Hendrix	I Don't Live Today	will i live tomorrow? \r well i just can't sa...
4997	Jennifer Lopez	Remember Me This Way	every now and then we find a special friend \...
4998	Cole Porter	Well Did You Evah!	have you heard? the coast of maine just got ca...
4999	Green Day	Still Breathing	[verse 1] \r i'm like a child looking off in ...

Tokenization

In [ ]:

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer

# Ensure the punkt tokenizer is downloaded
nltk.download('punkt')

# Initialize the stemmer
stemmer = PorterStemmer()

# Define the token function
def token(txt):
    tokens = word_tokenize(txt)
    a = [stemmer.stem(w) for w in tokens]
    return "  ".join(a)
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\PMLS\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

In [ ]:

```
token("you are beautiful, beauty")
```

Out[ ]:

```
'you  are  beauti  ,  beauti'
```

In [ ]:

```
df['text'].apply(lambda x:token(x))
```

```
Out[ ]: 0       there  are  river  run  just  for  you  and  m...
        1       hello  darlin  '  happi  birthday  i  've  dec...
        2       anoth  long  summer  's  come  and  gone  i  d...
        3       a  fine  romanc  ,  with  no  kiss  a  fine  r...
        4       look  at  the  rain  feel  the  pain  of  love...
                                       ...
        4995    raymond  versu  raymond  there  's  three  sid...
        4996    will  i  live  tomorrow  ?  well  i  just  ca ...
        4997    everi  now  and  then  we  find  a  special  f...
        4998    have  you  heard  ?  the  coast  of  main  jus...
        4999    [  vers  1  ]  i  'm  like  a  child  look  of...
        Name: text, Length: 5000, dtype: object
```

Feature Extraction

```
In [ ]: from sklearn.feature_extraction.text import TfidfVectorizer
        from sklearn.metrics.pairwise import cosine_similarity
```

```
In [ ]: tfid = TfidfVectorizer(analyzer='word', stop_words='english')
```

```
In [ ]: matrix = tfid.fit_transform(df['text'])
```

- **converting textual data into numerical so that we can apply ML algorithms**

```
In [ ]: similar = cosine_similarity(matrix)
```

```
In [ ]: similar[0]
```

```
Out[ ]: array([1.        , 0.03538886, 0.01560048, ..., 0.11059804, 0.00536044,
               0.03248334])
```

```
In [ ]: df[df['song']=="I Don't Live Today"].index[0]
```

```
Out[ ]: 4996
```

Building a Recommender Model

```
In [ ]: def recommender(song_name):
            # Check if the song exists in the DataFrame
            if song_name not in df['song'].values:
                return f"Song '{song_name}' not found in the dataset."

            # Get the index of the given song name
            idx = df[df['song'] == song_name].index[0]

            # Compute the distance and sort
            distance = sorted(list(enumerate(similar[idx])), reverse=True, key=lambda x: x[

            # Collect recommended songs
            recommended_songs = []
```

```
    for s_id in distance[1:6]:  # Get top 5 recommendations
        recommended_songs.append(df.iloc[s_id[0]].song)

    return recommended_songs
```

Results

In [ ]:

```
song_recommendation = recommender("Monstar")
print(song_recommendation)
```

```
['Little Lover', "I'll Be There For You", 'Cry To Me', "I Don't Want To Be Your Love
r", 'Lover Come Back To Me']
```