Music Recommendation

Project Goal:

The goal of this project is to build a music recommendation system that can recommend songs to users based on their listening history and preferences. The system will use a variety of factors to make recommendations, such as the genre, tempo, and mood of the songs that the user has listened to in the past.

Project Scope

The project will involve the following tasks:

1. Collecting a dataset of music tracks and their metadata.
2. Extracting features from the music tracks, such as the genre, tempo, and mood.
3. Developing a recommendation algorithm that can use the features to recommend songs to users.
4. Building a user interface for the recommendation system.

Importing Libraries

```
import numpy as np
import pandas as pd
```



```
from typing import List, Dict

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```

Dataset

```
songs = pd.read_csv('/content/songdata.csv')
```

```
songs.head()
```

	artist	song	link	
0	ABBA	Ahe's My Kind Of Girl	/a/abba/ahes+my+kind+of+girl_20598417.html	Look at her face, it's
1	ABBA	Andante, Andante	/a/abba/andante+andante_20002708.html	Take it easy with me, p
2	ABBA	As Good As New	/a/abba/as+good+as+new_20003033.html	I'll never know why I
3	ABBA	Bang	/a/abba/bang_20598415.html	Making somebody happ
4	ABBA	Bang-A-Boomerang	/a/abba/bang+a+boomerang_20002668.html	Making somebody happ

```
songs = songs.sample(n=5000).drop('link', axis=1).reset_index(drop=True)

songs['text'] = songs['text'].str.replace(r'\n', '')

tfidf = TfidfVectorizer(analyzer='word', stop_words='english')

lyrics_matrix = tfidf.fit_transform(songs['text'])

cosine_similarities = cosine_similarity(lyrics_matrix)
```

Saved successfully!

```
for i in range(len(cosine_similarities)):
    similar_indices = cosine_similarities[i].argsort()[:-50:-1]
    similarities[songs['song'].iloc[i]] = [(cosine_similarities[i][x], songs['song'][x], song

class ContentBasedRecommender:
    def __init__(self, matrix):
        self.matrix_similar = matrix

    def _print_message(self, song, recom_song):
        rec_items = len(recom_song)

        print(f'The {rec_items} recommended songs for {song} are:')
        for i in range(rec_items):
```

```
            print(f"Number {i+1}:")
            print(f"{recom_song[i][1]} by {recom_song[i][2]} with {round(recom_song[i][0], 3)
            print("--------------------")

    def recommend(self, recommendation):
        song = recommendation['song']
        number_songs = recommendation['number_songs']
        recom_song = self.matrix_similar[song][:number_songs]
        self._print_message(song=song, recom_song=recom_song)


recommedations = ContentBasedRecommender(similarities)


recommendation = {
    "song": songs['song'].iloc[10],
    "number_songs": 4
}


recommedations.recommend(recommendation)
```

```
 The 4 recommended songs for Peyton Place are:
 Number 1:
 A Face Like That by Pet Shop Boys with 0.13 similarity score
 --------------------
 Number 2:
 My Secret Place by Joni Mitchell with 0.124 similarity score
 --------------------
 Number 3:
 In Another Place And Time by Donna Summer with 0.12 similarity score
 --------------------
 Number 4:
 Face To Face by Reba Mcentire with 0.118 similarity score
 --------------------
```

Saved successfully!

```
    "number_songs": 4
}


recommedations.recommend(recommendation2)
```

```
 The 4 recommended songs for On My Way are:
 Number 1:
 Man In The Moon by Yes with 0.275 similarity score
 --------------------
 Number 2:
 Don't Worry, Kyoko by Yoko Ono with 0.228 similarity score
 --------------------
 Number 3:
 It Was Love That We Needed by Rod Stewart with 0.217 similarity score
 --------------------
```

```
Number 4:
Too Late by Journey with 0.19 similarity score
--------------------
```