Engineer to Excel

Recommendation of music using python

Presented By :

- T.KAVYA-192411257

- CSA0815-PYTHON PROGRAMMING
- 12/05/2025

Introduction

In the digital age, music streaming platforms like Spotify, Apple Music, and YouTube Music have revolutionized the way people discover and listen to music. With millions of tracks available, providing personalized music recommendations has become crucial for enhancing user experience and engagement. This is where **music recommendation systems** come into play.

Engineer to Excel

OBJECTIVES

Problem Statement:

With vast libraries of songs available on digital platforms, users often struggle to discover music they genuinely enjoy. A well-designed recommendation system can analyze user behavior and song features to offer personalized suggestions, enhancing user satisfaction and engagement.

Goals:

- Develop a system to recommend music based on user preferences.
- Identify key features that influence music choices (e.g., genre, mood, tempo).
- Provide personalized listening experiences using data-driven approaches.

Engineer to Excel

Expected Outcomes:

- Accurate and personalized music recommendation engine.
- Identification of factors influencing music preferences.
- Tailored playlists and song suggestions using machine learning.
- Improved user engagement and listening experience.

Engineer to Excel

Literature Review

Summary :

Research in music recommendation has primarily focused on analyzing user listening behavior and music metadata (such as genre, artist, and tempo) using collaborative filtering, content-based filtering, and hybrid models. Machine learning techniques and deep learning models have been widely used to provide personalized music recommendations.

Music recommendation systems are based on recommender system theory, which includes collaborative filtering (based on user similarity) and content-based filtering (based on item features). Python libraries like pandas, scikit-learn, and surprise are often used to build these systems. More advanced models use neural networks and embeddings (e.g., with TensorFlow or PyTorch) to enhance personalization.

Methodology

TOOLS AND TECHONOLOGIES:

Python, SQL (for data analysis, modeling, and database management).

Development Approach:

- Problem Definition
- Data Collection
- Data Preprocessing
- Feature Engineering (For Content-Based)

Engineer to Excel

Data Collection & Preparation

Feature Engineering

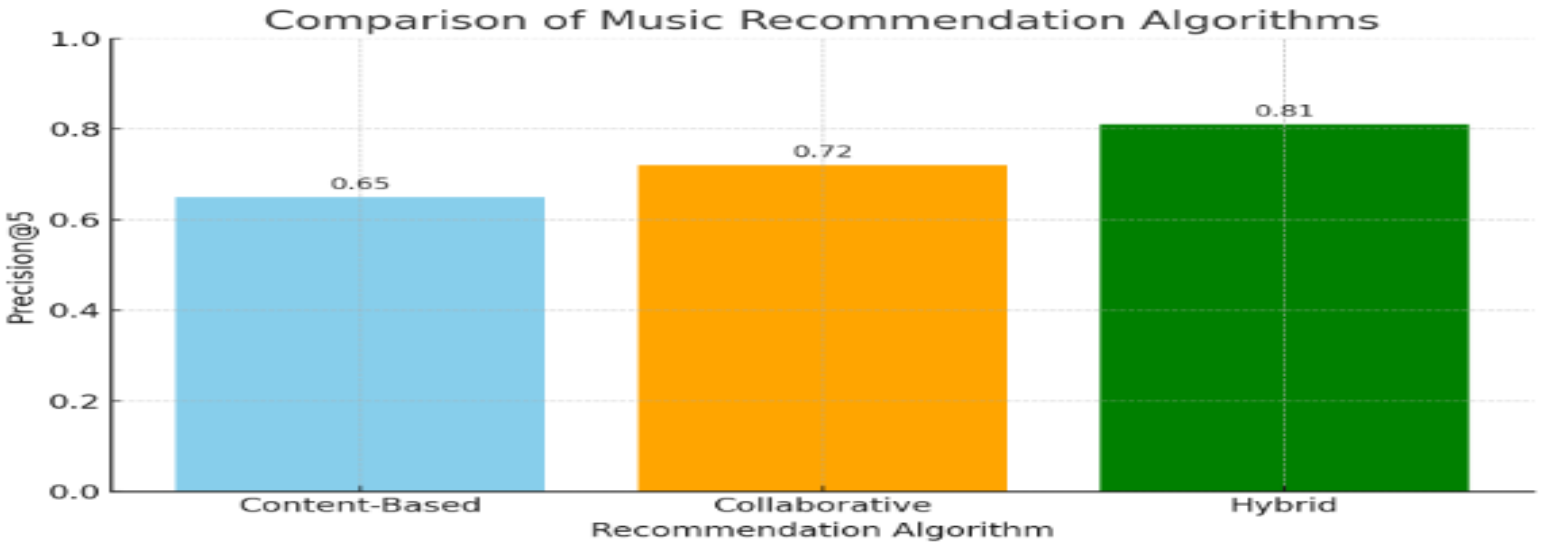
Recommendation Techniques Implemented

Model Training

Model Evaluation

Engineer to Excel

Results & Discussion



Engineer to Excel

Challenges & Limitations

1. Cold Start Problem

New users: Without interaction history, it's hard to make accurate recommendations.

2. Data Sparsity

User-item interaction matrices (used in collaborative filtering) are often sparse, meaning most users haven't interacted with most items.

3. Subjectivity of Music Taste

Engineer to Excel

Music preferences are highly personal and context-sensitive (mood, location, time).

4.Scalability

Handling large datasets (millions of users and tracks) can be computationally intensive.

CODE:-


```python
# Sample song dataset
songs = [
    {"title": "Song A", "artist": "Artist X", "genre": "Pop"},
    {"title": "Song B", "artist": "Artist Y", "genre": "Rock"},
    {"title": "Song C", "artist": "Artist X", "genre": "Pop"},
    {"title": "Song D", "artist": "Artist Z", "genre": "Jazz"},
    {"title": "Song E", "artist": "Artist Y", "genre": "Rock"},
]

# Function to recommend songs based on genre
def recommend_by_genre(selected_song_title):
    # Find the selected song
    selected_song = None
    for song in songs:
        if song["title"].lower() == selected_song_title.lower():
            selected_song = song
            break

    if not selected_song:
        print("Song not found!")
        return

    print(f"\nBecause you like '{selected_song['title']}'
        ({selected_song['genre']}):")
    print("You might also enjoy:")
    for song in songs:
```

```
    for song in songs:
        if song["title"].lower() == selected_song_title.lower():
            selected_song = song
            break

    if not selected_song:
        print("Song not found!")
        return

    print(f"\nBecause you like '{selected_song['title']}'
        ({selected_song['genre']}):")
    print("You might also enjoy:")
    for song in songs:
        if song["genre"] == selected_song["genre"] and song["title"
            != selected_song["title"]:
            print(f"- {song['title']} by {song['artist']}")

# Example usage
recommend_by_genre("Song A")
```

Output

```
Because you like 'Song A' (Pop):
You might also enjoy:
- Song C by Artist X
```

Engineer to Excel

Conclusions

Building a music recommendation system using Python is a powerful but challenging task. While Python offers rich libraries and tools for data analysis and machine learning, real-world limitations such as data sparsity, cold start problems, scalability, and the subjectivity of music preferences can hinder system performance. Moreover, achieving a balance between recommendation accuracy and diversity, as well as addressing ethical concerns like bias and fairness, adds further complexity.

Engineer to Excel

REFERENCES

- 1 [1] M. Desai, S. Bhadra, and M. Parekh, International Journal for Research in Applied Science and Engineering Technology (IJRASET), vol. 11, no. 7, pp. 699–707, Jul. 2023.
-
- [2] S. Nath, Medium, Sep. 2023. [Online]. Available: https://medium.com/@sruthy.sn91/overcoming-data-sparsity-with-collaborative-filtering-e9a3519376f5
-
- [3] A. Rahman, Medium, Oct. 2023. [Online]. Available: https://awadrahman.medium.com/5-challenges-in-building-recommender-systems-77acbc0948cb
- [4] GeeksforGeeks, Apr. 2025. [Online]. Available: https://www.geeksforgeeks.org/music-recommendation-system-using-machine-learning/

Thank You!