ADVENTIST UNIVERSITY OF CENTRAL AFRICA

Data Mining - Group 4

ID	Name
100883	Pierre Celestin Niyomugabo
100909	Emerithe Dusabimana
101002	Justin Tuyisenge
100912	JABO Jean Josue
100905	Herve Kagimbura

Project name: **"Rwandan Music Recommendation System"**

---

1 Introduction

In this project, we implemented a hybrid music recommendation system to Rwandan music. The system integrates unsupervised learning techniques and user listening history to suggest songs to users. The recommendation engine is intended to promote a broad range of Rwandan musical genres including Afropop, RnB, Gospel, Karahanyuze, Traditional, and Hip-hop.

Additionally, we developed as a lightweight web application using Flask and HTML to enable user check recommended musics.

2 Dataset Construction

The music dataset was manually curated from a selection of songs popular in Rwanda. It includes metadata such as Title, Artist, Genre, Album, and Year. The dataset features artists like Bruce Melodie, Meddy, The Ben, Ariel Wayz, Mike Kayihura, Israel Mbonyi, and traditional icons like Masamba Intore and Jean Paul Samputu. Each entry is assigned a unique TrackID.

3 Feature Engineering

To prepare the data for clustering and similarity analysis, categorical features such as Genre and Artist were encoded using LabelEncoder. These features, along with the song release year, were standardized using the StandardScaler to normalize the input for clustering and similarity measures.

```
features = music_df.copy()
features['GenreCode'] = LabelEncoder().fit_transform(features['Genre'])
features['ArtistCode'] = LabelEncoder().fit_transform(features['Artist'])
X = features[['GenreCode', 'ArtistCode', 'Year']]
X_scaled = StandardScaler().fit_transform(X)
```

4 Clustering and Similarity Computation

We used KMeans clustering algorithm to group songs into thematic or stylistic clusters. The optimal number of clusters is determined using the Elbow Method with KneeLocator, which analyzes the inertia values of different K values to find the elbow point. Cosine similarity is then computed on the standardized feature matrix to measure how similar each song is to others in the dataset.

```
def get_best_k(X):
    distortions = []
    for k in range(1, 10):
        km = KMeans(n_clusters=k, random_state=42).fit(X)
        distortions.append(km.inertia_)
    kl = KneeLocator(range(1, 10), distortions, curve="convex", direction="decreasing")
    return kl.elbow or 3

k = get_best_k(X_scaled)
```

5 User Interaction and Recommendation Logic

We implemented Flask web interface to allow users to search and select a song by title. Upon selection, the system retrieves the most similar songs using the cosine similarity scores. The implementation offers content-based filtering, and implicit user profiling through session-based listening history.

```
def get_recommendations(track_id, top_n=3):
    if track_id not in trackid_to_index:
        return []

    # Content-based
    idx = trackid_to_index[track_id]
    content_scores = list(enumerate(cos_sim[idx]))
    content_scores = sorted(content_scores, key=lambda x: x[1], reverse=True)[1:top_n+1]
    content_recs = [features.iloc[i]["TrackID"] for i, _ in content_scores]

    # Collaborative-based
    if track_id in user_item_matrix.columns:
        users = user_item_matrix[user_item_matrix[track_id] > 0].index
        subset = user_item_matrix.loc[users]
        co_scores = subset.drop(columns=track_id).sum().sort_values(ascending=False)
        collab_recs = co_scores.head(top_n).index.tolist()
    else:
        collab_recs = []

    # Merge (prioritize intersection, then union)
    hybrid = list(dict.fromkeys(content_recs + collab_recs))[:top_n]
    return music_df[music_df["TrackID"].isin(hybrid)].to_dict(orient="records")
```

6 Repo

The codes can be found from this Git Repo: https://github.com/CelestinNiyomugabo/Python-Lab/tree/689104160329b83b407e0d10c16d9c29cbb2da84/Music%20Recommendation%20App

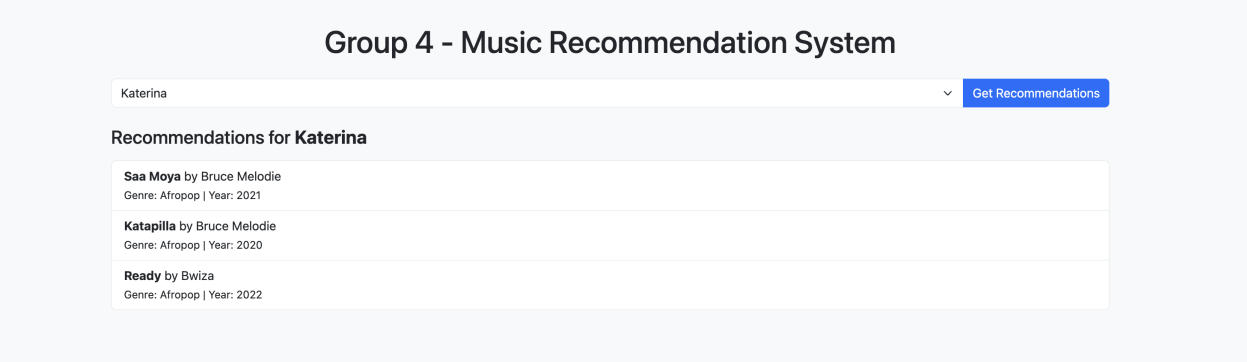

Figure 1: Dashboard