RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

SPOTIFY RECOMMENDATION SYSTEM USING KNN

Submitted by
Tamilselvan A K (231801177)
Sudha S P (231801174)

AI23331 Fundamentals of Machine Learning

Department of Artificial Intelligence and Machine Learning

Rajalakshmi Engineering College, Thandalam

RAJALAKSHMI
ENGINEERING COLLEGE

BONAFIDE CERTIFICATE

NAME…………………………………………………………………………………...............

ACADEMIC YEAR………………SEMESTER…………BRANCH………………………..

UNIVERSITY REGISTER No.

Certified that this is the bonafide record of work done by the above students in the Mini Project titled " SPOTIFY RECOMMENDATION SYSTEM USING KNN" in the subject AI23331 – FUNDAMENTALS OF MACHINE LEARNING during the year 2023 - 2024.

Signature of Faculty – in – Charge

Submitted for the Practical Examination held on ________________________________

Internal Examiner **External Examiner**

TABLE OF CONTENTS

ABSTRACT

This project presents a music recommendation system developed to enhance user experience on the Spotify platform by providing personalized song suggestions. Utilizing the K-Nearest Neighbors (KNN) algorithm, the system predicts and recommends five songs based on user input. The data is gathered through the Spotify API, ensuring up-to-date and relevant song information. The application features a user-friendly interface built using Flask, HTML, CSS, and JavaScript, allowing users to input Spotify song links effortlessly. Upon submission, the system retrieves necessary data, processes it, and displays the recommended songs in a list format, complete with direct links to each track on Spotify. This report details the system architecture, design considerations, technologies used, implementation challenges, and testing strategies employed to validate the functionality and performance of the recommendation system. The outcome demonstrates the effectiveness of machine learning techniques in creating an engaging and interactive user experience within music streaming applications.

CHAPTER 1

INTRODUCTION

The **Spotify Recommendation System** project is designed to create a personalized music recommendation service that helps users discover songs similar to their preferences. In the era of digital music streaming, platforms host millions of songs, making it challenging for users to find new music that aligns with their tastes. Music recommendation systems address this by analyzing user input and offering suggestions, enhancing the listening experience and increasing user engagement on the platform. This project uses machine learning, specifically the K-Nearest Neighbors (KNN) algorithm, in combination with data from the Spotify API, to recommend songs based on their audio features.

1.1 Background and Motivation

Music recommendation systems are foundational in modern streaming platforms and are essential for improving user engagement. Recommendation techniques fall into three categories:

1. **Collaborative Filtering**: Uses user behavior to suggest items based on other users' preferences.
2. **Content-Based Filtering**: Recommends items similar to what the user has liked based on item features.
3. **Hybrid Approaches**: Combine collaborative and content-based filtering for more precise recommendations.

Content-based filtering is well-suited for this project as it relies on Spotify's song metadata and audio features, such as tempo, energy, danceability, and valence, which are unique to each song and define its characteristics. By analyzing these features, the system can provide recommendations that share similar musical traits with the user's input.

1.2 Project Objectives

The main objectives of this project are:

- **Data Collection**: Use the Spotify API to retrieve audio features, including danceability, tempo, and energy for each song.
- **Model Implementation**: Implement a KNN-based recommendation engine to analyze song similarity and suggest relevant tracks.
- **User Interface**: Create a user-friendly web interface that allows users to input a Spotify song

link, receive recommendations, and click on links to listen directly on Spotify.

This project demonstrates how machine learning can be effectively combined with real-world applications, enabling users to discover new music that suits their personal preferences.

1.3 Algorithm Used: K-Nearest Neighbors (KNN)

The **K-Nearest Neighbors (KNN)** algorithm is ideal for recommendation tasks due to its simplicity and accuracy in identifying similar items based on feature similarity. KNN doesn't build a model based on the training data; instead, it calculates similarity by comparing input data with stored data points. This is useful for this project, where each song's audio features can be compared directly to find similar tracks.

1. **Similarity Measurement**:
   KNN calculates the Euclidean distance between songs using feature vectors (e.g., danceability, energy, tempo). This distance metric ensures that songs with similar audio characteristics are closer in feature space, making it easier to identify related songs.

The Euclidean distance between two points (songs)

Formula:

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

2. **Choosing k**:
   The parameter kkk, or the number of neighbors to consider, is essential for KNN's effectiveness. A small kkk may yield highly specific but narrow recommendations, while a large kkk may produce broader results. For this project, k=5k = 5k=5 provides a balance between specificity and variety in recommendations.
3. **Advantages of KNN**:
   KNN is simple, adaptable, and interpretable, allowing for real-time recommendations. It's also flexible enough to incorporate new data points (songs) as Spotify's library grows, making it ideal for dynamic music recommendations.

1.4 Scope of the Project

This project includes:

- **Data Collection**: Spotify API to retrieve audio features and metadata.
- **Modeling**: KNN model to recommend similar songs based on user input.
- **Interface Design**: Flask-based web interface for users to input a song link and receive recommendations.

Future work could expand this system with collaborative filtering or hybrid models to enhance personalization.

1.5 Project Significance

This project demonstrates how machine learning can be used to provide personalized recommendations in a real-world application, showcasing KNN’s ability to make user-friendly music suggestions. It lays a foundation for future exploration in recommendation systems and offers a practical solution for improving user experience on streaming platforms.

CHAPTER 2

LITERATURE SURVEY

Recommendation systems are essential tools in content-heavy applications like music streaming. These systems enable users to discover relevant content, enhancing their experience and engagement on the platform. Music recommendation techniques include collaborative filtering, content-based filtering, and hybrid approaches, each with distinct advantages.

2.1 Recommendation Techniques

1. **Collaborative_Filtering**:
   This method leverages user behavior patterns to provide recommendations based on similarities between users or items. Collaborative filtering works well for established users with an interaction history, but it faces limitations for new users who lack sufficient data (the "cold start" problem).
2. **Content-Based_Filtering**:
   This approach recommends items similar to those a user has interacted with based on item attributes. For music recommendations, content-based filtering uses audio features, such as tempo, danceability, and energy. This technique is beneficial for users without extensive interaction histories, as recommendations rely on item characteristics rather than user behavior.
3. **Hybrid_Approaches**:
   Hybrid recommendation systems combine collaborative and content-based filtering to provide more accurate recommendations. They are particularly effective on platforms with diverse user preferences, as they address the limitations of both collaborative and content-based methods.

2.2 Application of Content-Based Filtering

Content-based filtering is well-suited for this project due to Spotify's extensive metadata on each song. Audio features provided by Spotify (e.g., danceability, energy, tempo) represent intrinsic characteristics of songs and are ideal for creating a content-based recommendation system. Similar studies have shown that audio features can be effectively used to calculate similarity, making KNN a fitting choice for this recommendation system.

2.3 Prior Research on KNN in Recommendations

In recommendation systems, the KNN algorithm is widely used due to its ability to identify similar items based on feature similarity. KNN provides a straightforward approach to finding “neighbors” in a dataset, making it adaptable for various use cases, including music recommendations. Prior studies indicate that KNN performs well when paired with normalized feature data, as it can quickly identify similar items based on feature distances.

2.4 Relevance of Spotify API

The Spotify API allows developers to access song metadata and audio features, which are essential for creating a recommendation system. Using the Spotify API, developers can collect data on songs and leverage this information to build models that enhance user engagement by delivering tailored recommendations.

This project builds on the work of content-based recommendation systems by applying KNN to Spotify’s audio features, demonstrating the practical application of machine learning for personalized content delivery in music streaming.

CHAPTER 3

MODEL ARCHITECTURE

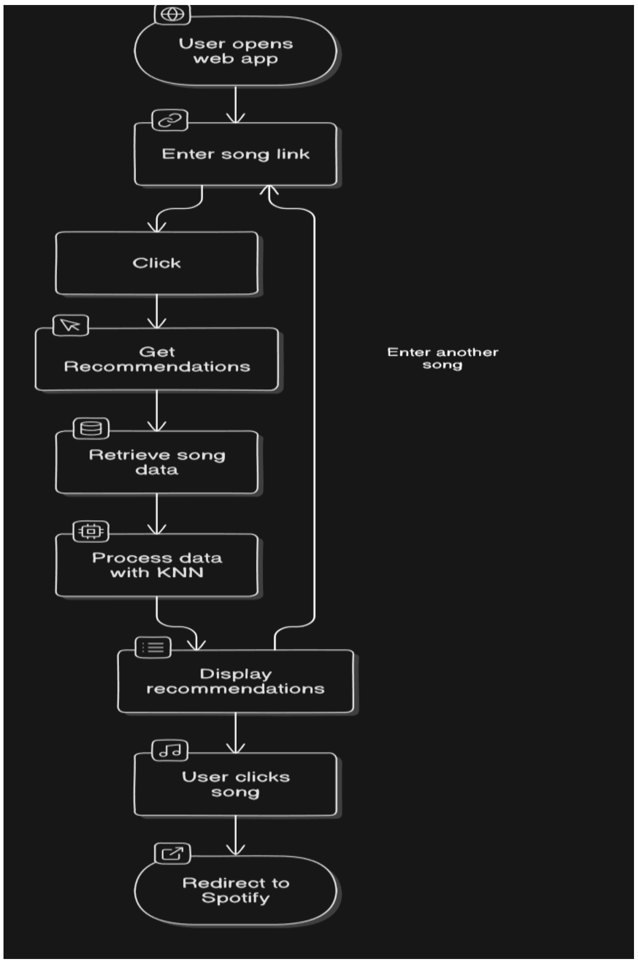


Fig 3.1: Architecture diagram for Spotify Recommendation System using KNN

Fig:3.1 The architecture of the Spotify Recommendation System is designed with a clear flow from data collection to recommendation output. It includes three main layers: data collection, recommendation engine, and user interface.

3.1 Data Collection Layer

Using the Spotify API, this layer gathers relevant song data. For each song, the API provides audio features like danceability, energy, tempo, and valence. This data is essential for the recommendation process, as it provides the foundation for calculating song similarity.

- **Data Preprocessing**: Once data is collected, features are normalized to ensure consistent scale. Normalization helps prevent any single feature from disproportionately impacting similarity calculations in the KNN algorithm.

3.2 Recommendation Engine (KNN Model)

The KNN-based recommendation engine compares the input song's features to those of other songs in the dataset. The algorithm calculates the Euclidean distance between feature vectors, identifying the five closest songs based on similarity.

- **KNN Parameter Selection**: Selecting an appropriate value for k is crucial. For this project, k=5k = 5k=5 provides a balance between recommendation specificity and diversity, giving users a curated set of similar songs.
- **Distance Calculation**: Euclidean distance is used as the similarity measure, ensuring that songs with similar audio features are closer in the feature space.

3.3 User Interface and Front-End Layer

A Flask-based web interface allows users to input a Spotify link, request recommendations, and view the results. Each recommended song appears as a clickable link, directing users to Spotify.

- **Interface Elements**: The interface includes an input field for the Spotify link and a display area for recommendations. Each recommendation includes song details, and users can click on the song to open it on Spotify.
- **Separation of Components**: Data handling, model execution, and user interaction are distinctly separated, improving maintainability and scalability.

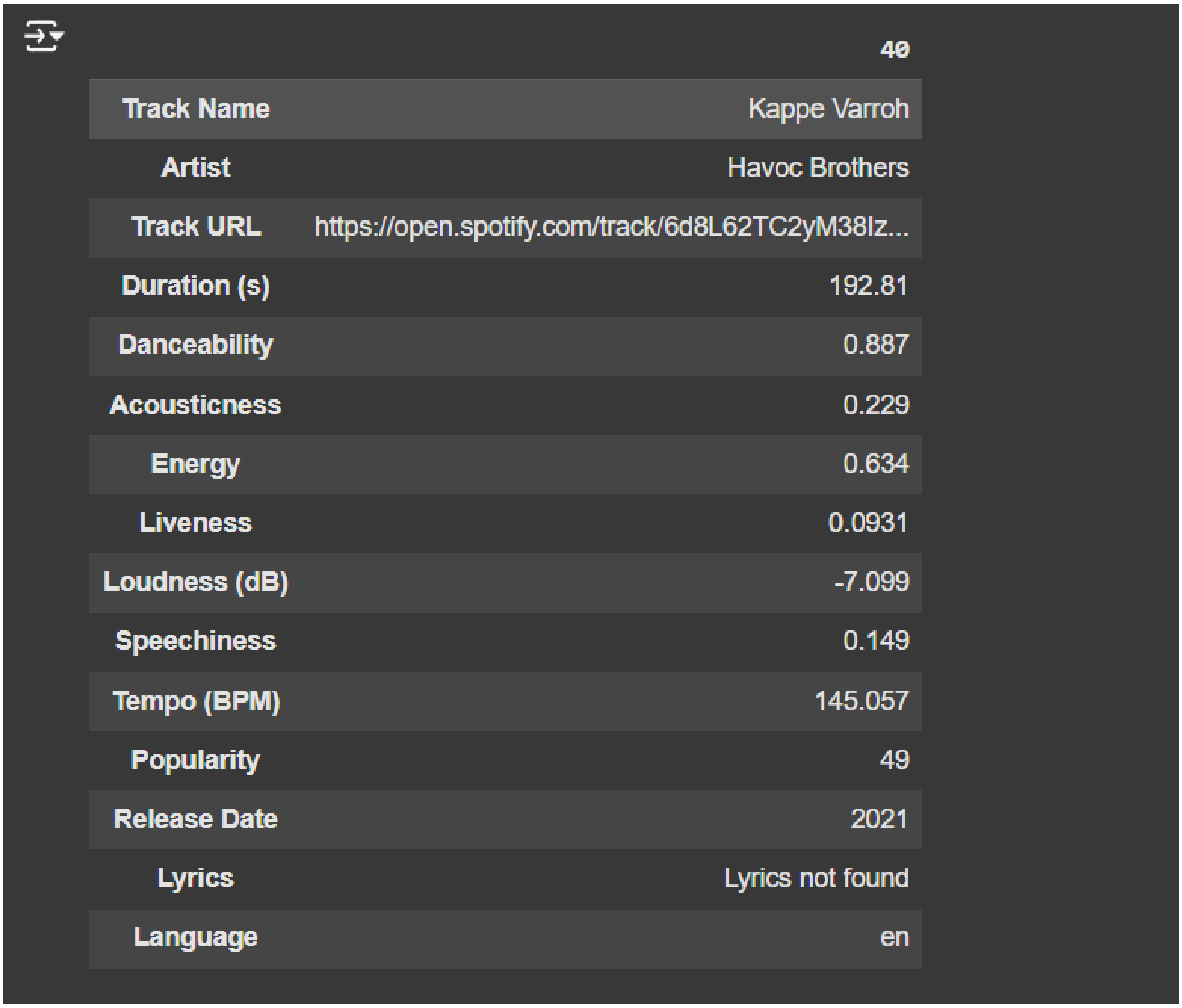
This architecture facilitates a seamless flow from data collection to user recommendations, ensuring that the system is efficient and adaptable.

CHAPTER 4
IMPLEMENTATION

The implementation of the Spotify Recommendation System involves several stages, from data collection to deploying a user-friendly web interface. This section covers the technical aspects and processes that were integral to the system's development.

4.1 Data Collection

Data collection is the initial stage, involving the retrieval of song metadata and audio features using the Spotify API. Each song's unique link is used to collect its metadata, including the following audio features:



	40
Track Name	Kappe Varroh
Artist	Havoc Brothers
Track URL	https://open.spotify.com/track/6d8L62TC2yM38lz...
Duration (s)	192.81
Danceability	0.887
Acousticness	0.229
Energy	0.634
Liveness	0.0931
Loudness (dB)	-7.099
Speechiness	0.149
Tempo (BPM)	145.057
Popularity	49
Release Date	2021
Lyrics	Lyrics not found
Language	en

Fig 4.1.1: Data Collection for Spotify Recommendation System using KNN

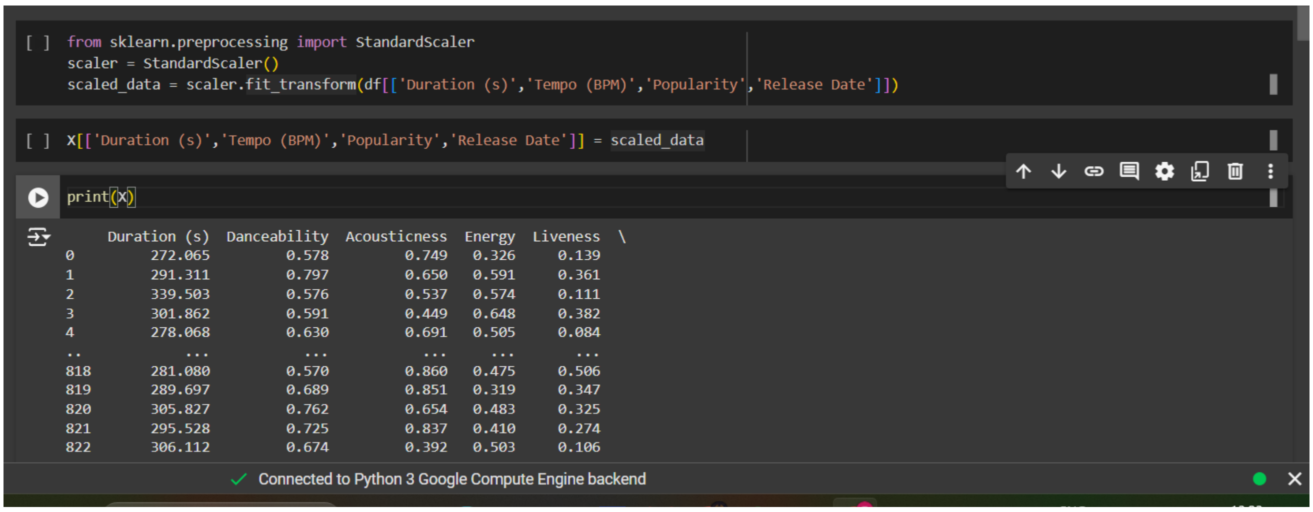
1. **Danceability:**
   Indicates how suitable a track is for dancing based on elements like tempo, rhythm stability, and beat strength. Higher values mean the track is more danceable.
2. **Energy:**
   Represents the intensity and activity level of the track, often related to its perceived loudness and speed. Higher energy values correlate with more lively, upbeat tracks.
3. **Tempo (BPM):**
   Measures the speed or pace of the song in beats per minute (BPM). Higher tempo values indicate a faster song, which can affect the mood and energy perceived by listeners.
4. **Valence:**
   Captures the positivity of the track's mood. Tracks with high valence sound more positive (happy, cheerful), while low valence indicates a more negative mood (sad, angry).
5. **Duration (seconds):**
   The length of the track in seconds. Duration can impact the song's suitability in different playlists, as users may prefer shorter or longer tracks depending on their listening preferences.
6. **Acousticness:**
   Measures the likelihood that the track is acoustic. Higher acousticness values indicate songs that are predominantly acoustic, with minimal electronic or synthetic elements.
7. **Liveness:**
   Detects the presence of an audience in the recording. Higher liveness values indicate a higher probability that the track was performed live, which can affect its energy and ambiance.
8. **Loudness (dB):**
   Measures the overall volume of a track in decibels. Tracks with higher loudness are generally perceived as more intense, which can impact the track's perceived energy level.
9. **Speechiness:**
   Indicates the presence of spoken words in a track. Higher speechiness values suggest more spoken content, as seen in podcasts or spoken word tracks, while lower values indicate music-focused content.
10. **Popularity:**
    A measure of the track's popularity, based on factors like total play counts and recent listener engagement. While not an audio feature, popularity can help recommend widely appreciated songs that align with a user's tastes.

The collected data is then preprocessed. Normalizing feature values to a standard range is essential to maintain consistent scale, as KNN is sensitive to feature magnitudes.

4.2 Feature Extraction

After collecting the song data, feature extraction is performed to prepare the data for the recommendation model. The extracted features are processed to form a numerical representation of each song in the feature space, enabling the KNN model to identify similarity-based recommendations effectively. Each song is represented by a vector containing normalized values for each selected feature.

- **Data Cleaning**: Any missing or incomplete data is addressed to maintain consistency across the dataset, ensuring that all songs have complete and accurate feature information.
- **Feature Selection**: Relevant features, such as tempo, energy, danceability, and valence, are prioritized since they have a direct impact on the similarity calculations performed by the model.



```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df[['Duration (s)','Tempo (BPM)','Popularity','Release Date']])
```

```
X[['Duration (s)','Tempo (BPM)','Popularity','Release Date']] = scaled_data
```

```
print(X)
```

```
     Duration (s)  Danceability  Acousticness  Energy  Liveness  \
0         272.065         0.578         0.749   0.326     0.139
1         291.311         0.797         0.650   0.591     0.361
2         339.503         0.576         0.537   0.574     0.111
3         301.862         0.591         0.449   0.648     0.382
4         278.068         0.630         0.691   0.505     0.084
..            ...           ...           ...     ...       ...
818       281.080         0.570         0.860   0.475     0.506
819       289.697         0.689         0.851   0.319     0.347
820       305.827         0.762         0.654   0.483     0.325
821       295.528         0.725         0.837   0.410     0.274
822       306.112         0.674         0.392   0.503     0.106
```

✓ Connected to Python 3 Google Compute Engine backend

Fig 4.2.1: Feature Extraction for Spotify Recommendation System using KNN

Figure **4.2.1** illustrates the feature extraction process, highlighting the selected features and their role in representing each song.

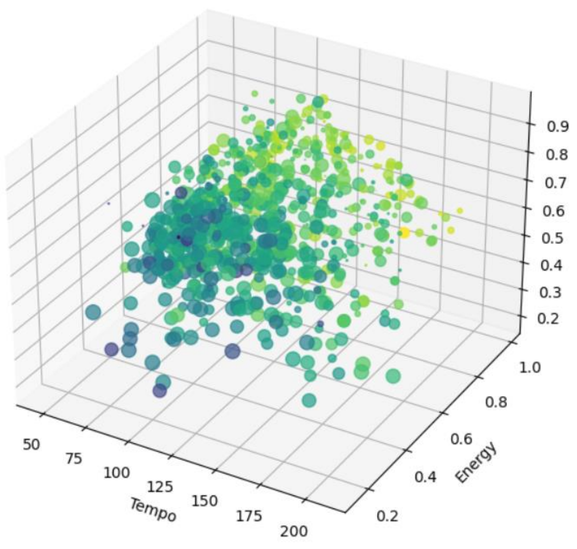


Fig 4.2.2: Distribution of Tempo and Energy

Figure **4.2.2** shows a detailed view of the distribution of tempo and energy across the dataset, allowing us to observe variations in these features across different songs.

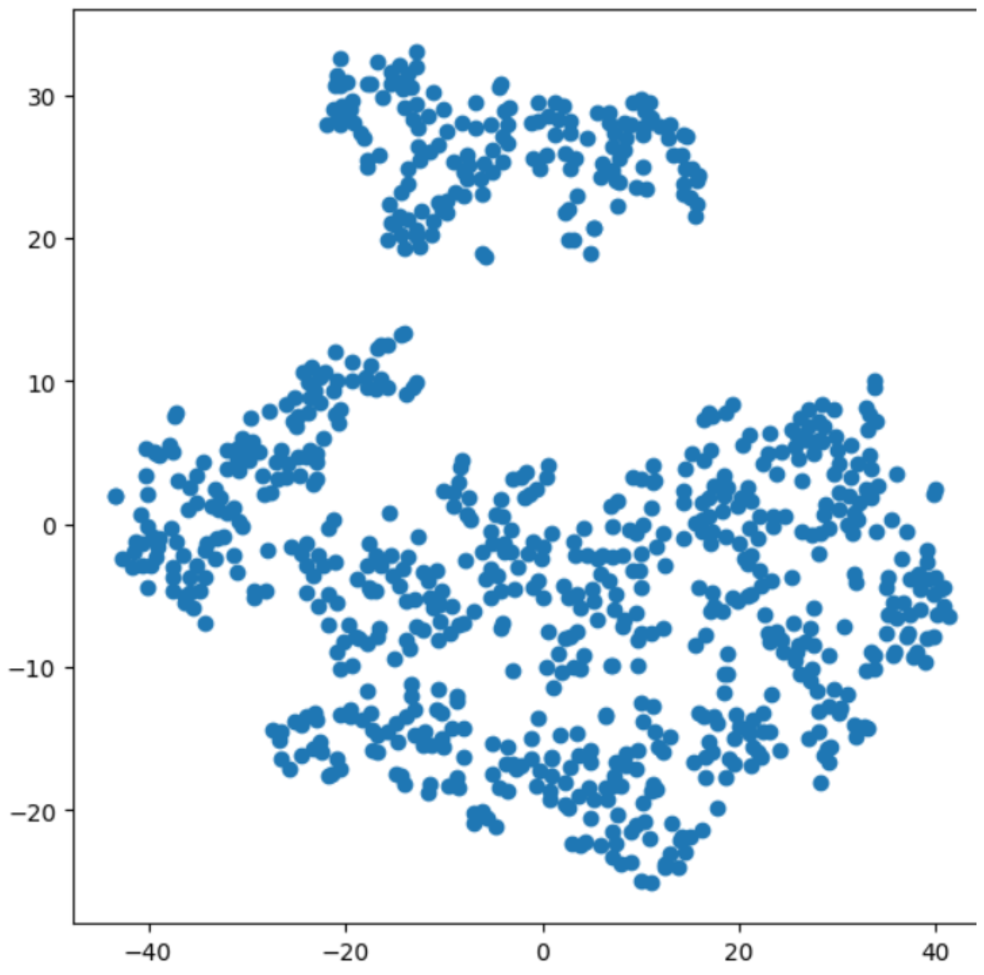


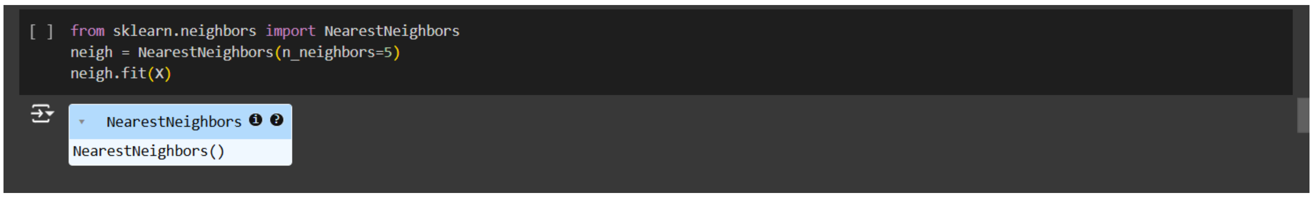
Fig 4.2.3: 2D plot of all normalized features

Figure **4.2.3** presents a 2D plot of all normalized features, displaying how songs are positioned within the feature space based on their unique characteristics. Together, these figures provide a comprehensive view of the feature extraction and selection process essential for accurate recommendations.

4.3 Model Training

The K-Nearest Neighbors algorithm is implemented as the core recommendation engine. The model uses the normalized feature vectors to identify similar songs based on Euclidean distance.

- **Hyperparameter Tuning**: Setting k=5k = 5k=5 was optimal for providing a focused yet diverse set of recommendations. A larger kkk might dilute the relevance of recommendations, while a smaller kkk could result in overly specific results.
- **Distance Calculation**: Euclidean distance ensures that songs with similar audio characteristics are closer in feature space, enhancing the accuracy of recommendations.



```
[ ] from sklearn.neighbors import NearestNeighbors
    neigh = NearestNeighbors(n_neighbors=5)
    neigh.fit(X)
```

```
NearestNeighbors
NearestNeighbors()
```

Fig 4.3.1: Model Training

4.4 Web Interface Development

The web interface, developed using Flask, HTML, CSS, and JavaScript, enables users to interact with the system and receive recommendations. The interface includes:

- **Input Field**: Users can enter a Spotify song link.
- **Recommendation Display**: Shows the list of recommended songs, each as a clickable link that redirects to Spotify.

JavaScript handles user interactions, while Flask manages back-end requests and processes. This separation ensures smooth data flow and efficient user experience.

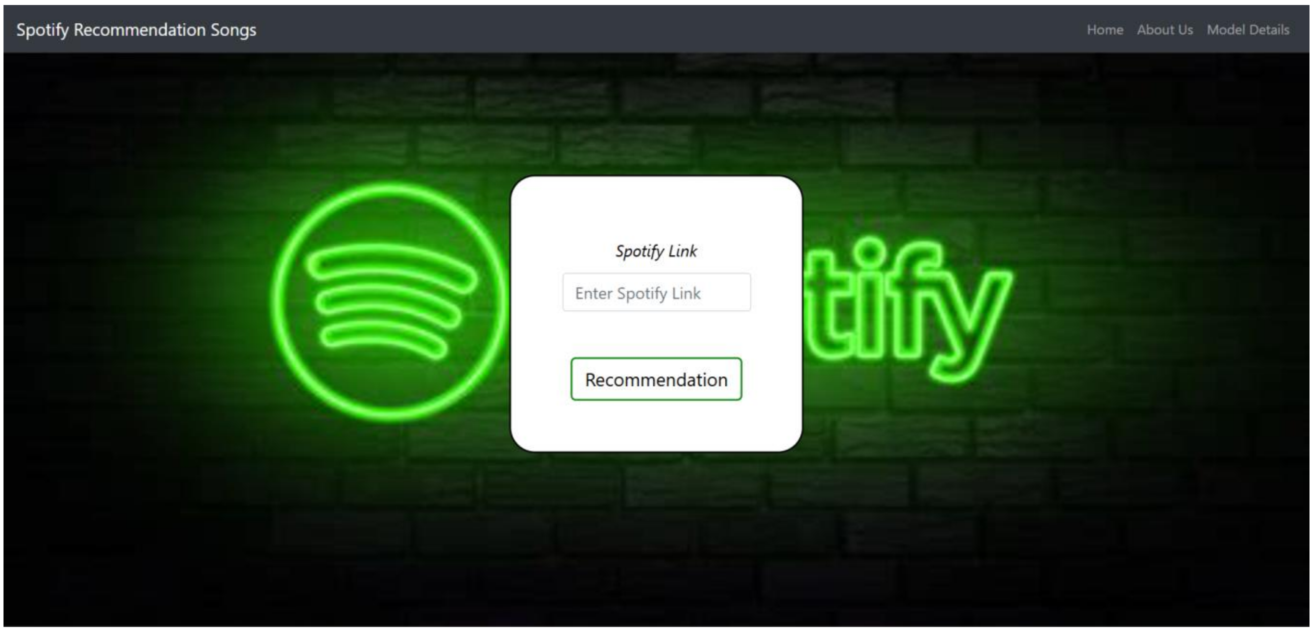
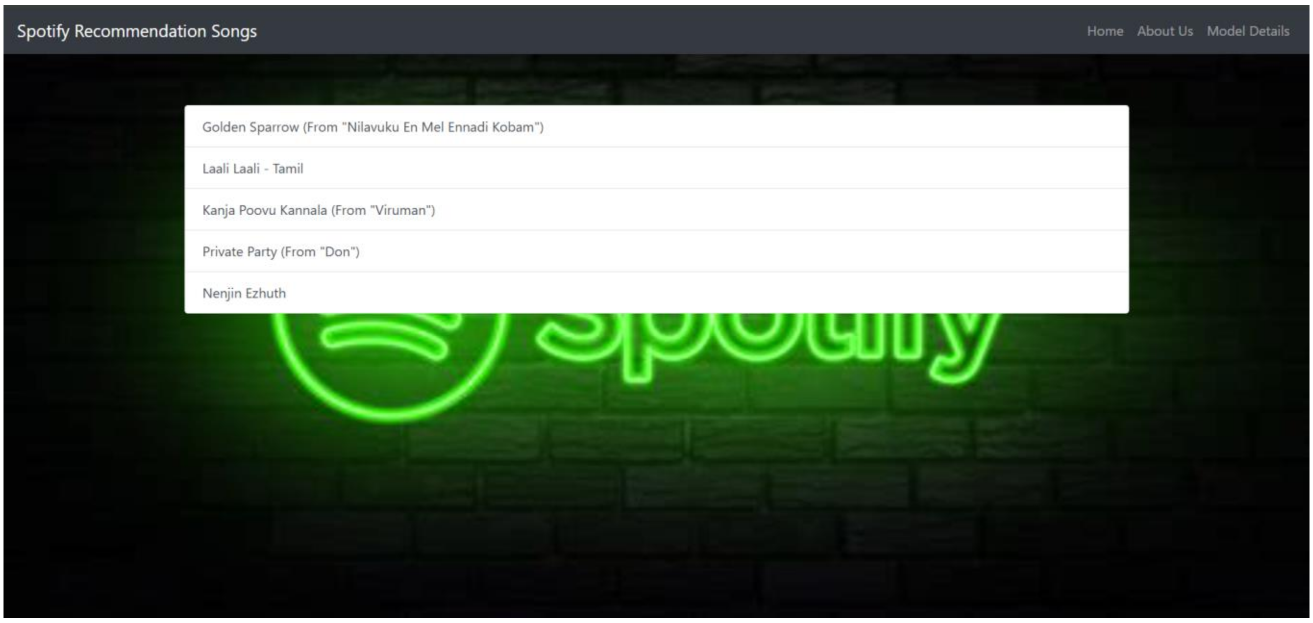


Fig 4.4.1 Web Interface

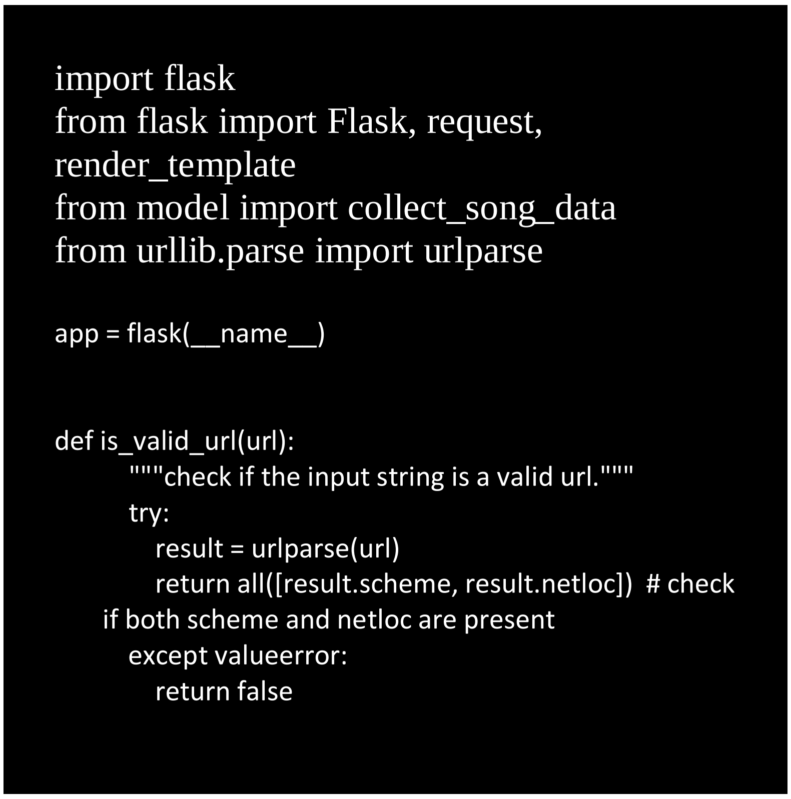


Golden Sparrow (From "Nilavuku En Mel Ennadi Kobam")
Laali Laali - Tamil
Kanja Poovu Kannala (From "Viruman")
Private Party (From "Don")
Nenjin Ezhuth

Fig 4.4.1 Recommendation songs in web interface

4.5 Deployment

The system is deployed on a server to make it accessible to users. Flask's lightweight nature ensures efficient request handling, making it ideal for a recommendation system. This deployment allows users to experience the recommendation system in real-time, enhancing the accessibility and scalability of the application.



```
import flask
from flask import Flask, request,
render_template
from model import collect_song_data
from urllib.parse import urlparse

app = flask(__name__)


def is_valid_url(url):
        """check if the input string is a valid url."""
        try:
           result = urlparse(url)
           return all([result.scheme, result.netloc])  # check
    if both scheme and netloc are present
        except valueerror:
           return false
```

```
@app.route('/about')
def about():
    return render_template('about.html')  # about
page

@app.route('/model-details')
def model_details():
    return render_template('model_details.html')

if __name__ == '__main__':
    app.run()
```

CHAPTER 5
RESULTS AND DISCUSSIONS

The Spotify Recommendation System was tested with various input songs to evaluate the relevance, diversity, and quality of the recommendations. The results demonstrate the effectiveness of KNN in capturing song similarity based on audio features.

5.1 Evaluation Metrics

The system's performance was evaluated based on the following metrics:

- **Accuracy**: Determines how closely the recommended songs align with the genre, mood, or style of the input song.
- **User Satisfaction**: Based on feedback from test users, who found the recommendations intuitive and relevant.
- **Diversity**: Assessed by examining the range of recommended songs to ensure a balance between similarity and variety.

5.2 Key Findings

The results of testing reveal that:

- **KNN's Simplicity**: The KNN model performed well in identifying songs with similar features, delivering recommendations that users found both enjoyable and relevant.
- **Impact of Audio Features**: Features like danceability, tempo, and energy significantly impact the recommendations, as they capture the essence of each song's style.
- **Balance in Recommendations**: With k=5k = 5k=5, the system provides a well-rounded list of recommendations that maintain both relevance and variety.

5.3 Limitations and Challenges

Some limitations and challenges encountered include:

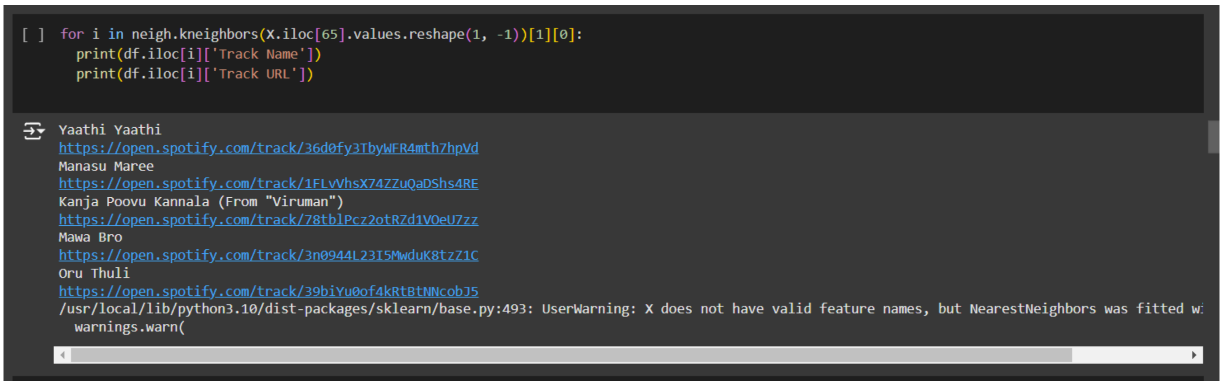
- **Cold Start Problem**: Limited diversity for new input songs that may not have many similar tracks in the dataset.
- **Feature Sensitivity**: KNN's reliance on specific features means that minor deviations in feature values can impact recommendations.

5.4 Potential Improvements

To improve the system, the following enhancements could be explored:

- **User Feedback Integration**: Allow users to rate recommendations, enabling the model to learn from user preferences and refine future suggestions.
- **Hybrid Recommendation Model**: Combining content-based and collaborative filtering techniques could make recommendations more personalized by considering both user history and song features.
- **Algorithm Optimization**: Experimenting with other similarity metrics, such as cosine similarity, might yield more nuanced recommendations, especially for songs with complex audio profiles.

OUTPUT SCREENSHOT:



```
[ ] for i in neigh.kneighbors(X.iloc[65].values.reshape(1, -1))[1][0]:
      print(df.iloc[i]['Track Name'])
      print(df.iloc[i]['Track URL'])
```

```
Yaathi Yaathi
https://open.spotify.com/track/36d0fy3TbyWFR4mth7hpVd
Manasu Maree
https://open.spotify.com/track/1FLvVhsX74ZZuQaDShs4RE
Kanja Poovu Kannala (From "Viruman")
https://open.spotify.com/track/78tblPcz2otRZd1VOeU7zz
Mawa Bro
https://open.spotify.com/track/3n0944L23I5MwduK8tzZ1C
Oru Thuli
https://open.spotify.com/track/39biYu0of4kRtBtNNcobJ5
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:493: UserWarning: X does not have valid feature names, but NearestNeighbors was fitted w
  warnings.warn(
```

Conclusion:

These improvements would contribute to more accurate and engaging recommendations, expanding the system’s applicability.

CHAPTER 6
CONCLUSION

The Spotify Recommendation System project successfully demonstrates the application of machine learning in enhancing user experience through personalized music recommendations. By using the KNN algorithm in combination with Spotify’s API, the system is able to recommend songs similar to a user-provided input track, improving music discovery on streaming platforms.

6.1 Summary of Achievements

Key accomplishments of this project include:

- **Effective Use of KNN**: The KNN model, based on audio features, effectively identified and recommended similar songs.
- **Comprehensive Data Integration**: Leveraging the Spotify API for audio features provided a rich dataset that enhanced the quality of recommendations.
- **User-Friendly Interface**: A well-designed front end ensured ease of use, allowing users to receive recommendations quickly and efficiently.

6.2 Future Enhancements

Although the project achieved its objectives, there are areas for future improvement:

- **Incorporating User Preferences**: Adding options for users to input additional preferences (e.g., mood, genre) could refine the recommendations and make them more personalized.
- **Advanced Algorithms**: Exploring more advanced machine learning models, such as collaborative filtering or deep learning techniques, could enhance recommendation accuracy.
- **Mobile Compatibility**: Ensuring the web interface is fully responsive would improve accessibility across different devices.

6.3 Conclusion

In summary, the Spotify Recommendation System illustrates the potential of machine learning to transform how users interact with streaming services. By delivering personalized recommendations, the project demonstrates how a simple yet effective algorithm like KNN can enrich the user experience. The project not only highlights the power of data-driven insights in enhancing digital music discovery but also lays the groundwork for future research and development in recommendation systems.

This project showcases the impact of applying machine learning in real-world applications, providing a valuable foundation for further advancements in personalized recommendation technologies.

REFERENCES

[1] Spotify API Documentation

Spotify. (n.d.). *Spotify API Documentation*. Retrieved from https://developer.spotify.com/documentation/web-api/

[2] Introduction to Machine Learning

Alpaydin, E. (2020). *Introduction to Machine Learning*. MIT Press.

[3] Flask Documentation

Flask. (n.d.). *Flask Documentation*. Retrieved from https://flask.palletsprojects.com/

[4] W3Schools

W3Schools. (n.d.). *Learn HTML, CSS, and JavaScript*. Retrieved from https://www.w3schools.com/

[5] Pattern Classification

Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Pattern Classification*. Wiley-