Fastai Music Recommender System with LSTMs and PCA-based Feature Reduction

Likhitha Alle
Dept. of Computer Engineering and Computer Science, California State University,
Long Beach, United States.
Likhitha.alle01@student.csulb.edu

Swetha Sri Guttula
Dept. of Computer Engineering and Computer Science, California State University,
Long Beach, United States.
SwethaSri.Guttula01@student.csulb.edu

***Abstract*—Our project aims to develop a comprehensive music recommendation system using the Spotify and User History dataset, providing users with personalized song recommendations based on their individual interests. By integrating data analysis, visualization, and advanced recommendation algorithms, we seek to offer users a deep understanding of their musical preferences. The system leverages Long Short-Term Memory (LSTM) networks to capture the sequential nature of music listening behavior for more accurate recommendations and utilizes Principal Component Analysis (PCA) for dimensionality reduction to enhance model efficiency and reduce overfitting. Fastai, a high-level deep learning library, is employed to streamline the development process, making the construction of complex architectures like LSTMs more accessible. The system's performance will be evaluated using standard metrics such as precision and recall, and compared against baseline models to assess the effectiveness of the proposed approach. This project provides valuable insights into the use of advanced machine learning techniques in creating accurate and personalized music recommender systems, benefiting researchers and practitioners in the field..**

***Keywords*:** fastai, Long Short-Term Memory(LSTM), Principal component analysis(PCA)

I. Introduction

Music recommender systems have become an essential tool for music streaming services like Spotify and Apple Music, helping users discover new music they might enjoy. These systems employ various techniques to analyze user listening habits and recommend songs or artists that align with their preferences. This report explores a music recommender system built using Fastai, a deep learning library. The system leverages Long Short-Term Memory (LSTM) networks, a type of recurrent neural network (RNN) adept at handling sequential data, to capture the nuances of music listening patterns. Additionally, the system incorporates Principal Component Analysis (PCA) for dimensionality reduction, which helps to improve the efficiency and accuracy of the recommendation process. By combining the strengths of Fastai, LSTMs, and PCA, this music recommender system aims to provide users with personalized and relevant music suggestions, enhancing their overall music listening experience.

A. How to run a code?

Upload Dataset: Begin by uploading the "Spotify Dataset" into your Google Drive account

- Run PR pro fastai File: Open the "PR pro fastai" file in Google Colab. Execute the code within the file to conduct data exploration and visualization using the Fastai library for creating a model in machine learning and music recommendation.
- Execute PR pro LSTM File: Next, open the "PR pro LSTM" file in Google Colab. Run the code provided to train a collaborative filtering model based on the Long Short-Term Memory (LSTM) technique for music recommendation.
- Run PR pro PCA File: Similarly, open the "PR pro PCA" file in Google Colab. Execute the code segments within the file to train a collaborative filtering model based on the Principal Component Analysis (PCA) technique for music recommendation.
- Follow Code Segments: Each file contains code segments for specific tasks, including data preprocessing, model training, and recommenda-

tion generation. Follow the code instructions within each file to ensure seamless execution of the machine learning models.
- Completion: By following these step by step instructions, users can effectively run the provided machine learning models for music recommendation using Google Colab.

Colab Link:

- PR project Fastai ipynb file link: Fastai Link
- PR project LSTM ipynb file link: LSTM Link
- PR project PCA ipynb file link: PCA Link

II. Background

In the digital age, music streaming services have revolutionized how people discover and enjoy music. Platforms like Spotify have amassed vast amounts of user data, including listening histories, preferences, and behavioral patterns. Leveraging this data to provide personalized music recommendations has become a critical component of enhancing user experience and engagement. Traditional recommendation systems, such as collaborative filtering and content-based filtering, have been widely used. Collaborative filtering relies on the collective preferences of many users to suggest new music, while content-based filtering focuses on the characteristics of the music itself to make recommendations. However, these methods often face challenges such as scalability, cold-start problems, and the inability to capture sequential user behavior effectively.

Scalability is a significant issue because the volume of data generated by millions of users can be overwhelming, requiring immense computational resources to process and analyze in real-time. The cold-start problem arises when new users or new music items are added to the system, making it difficult to generate accurate recommendations due to the lack of historical data. Moreover, traditional methods struggle to capture the sequential nature of music consumption, where the order and context of listened tracks play a crucial role in shaping user preferences. For example, a user might prefer upbeat songs in the morning and more relaxing tunes in the evening, a pattern that is not easily detected by conventional approaches.

III. Problem Statement

The primary challenge in music recommendation systems is predicting user preferences accurately and offering personalized recommendations that align with individual tastes. Traditional methods often fail to capture the temporal dynamics of user listening behavior and struggle with the high-dimensional nature of user-item interaction data. This project tackles these issues by incorporating advanced machine learning techniques, namely Long Short-Term Memory (LSTM) networks and Principal Component Analysis (PCA). LSTMs are particularly adept at modeling the sequential nature of music listening, allowing the system to continually learn and adapt to changes in user preferences. They also address the cold-start problem by using contextual information and metadata, enabling the system to provide accurate recommendations even with limited historical data. Additionally, LSTM networks enhance scalability by efficiently processing large datasets and managing high volumes of streaming data.

PCA complements LSTMs by reducing the dimensionality of the dataset, simplifying computational requirements, and improving system scalability. By transforming user and item features into principal components, PCA retains the essential patterns in the data while making it more manageable. This technique also helps address the cold-start problem by finding similarities between new users or songs and existing principal components, allowing the system to make informed predictions about preferences. Furthermore, PCA enables better integration of collaborative filtering and content-based methods, boosting overall recommendation performance. The combination of LSTM networks and PCA creates a powerful hybrid approach, leveraging the strengths of both methods to develop a more accurate, dynamic, and efficient music recommendation system. This integration enhances user satisfaction and engagement, showcasing the potential of advanced machine learning techniques to improve music streaming services.

IV. Objectives

- Analyze and visualize user listening histories to gain insights into individual musical prefer-

ences.the Long Short-Term Memory (LSTM) technique for music recommendation.
- Implement LSTM networks to capture sequential user behavior in music listening history, thereby improving the accuracy of recommendations.
- Apply PCA for dimensionality reduction of user-item interaction data to enhance model efficiency and reduce overfitting.
- By leveraging Fastai, music streaming services can efficiently implement and optimize advanced recommendation techniques like LSTMs and PCA, leading to more personalized, dynamic, and scalable recommendations that enhance user satisfaction and engagement.
- Evaluate the system's performance using standard metrics such as precision and recall, and compare it to baseline models to assess the overall effectiveness of the proposed system.

V. Dataset

The Spotify dataset used in this project is a comprehensive collection of musical tracks, encompassing a wide range of audio features and metadata. This dataset is sourced from Spotify's Web API and includes detailed information on over 50,683 songs spanning from 1921 to 2020. The dataset is available in CSV format, making it easy to load and manipulate using various data analysis tools.

- track id: A unique identifier for each track in the dataset.
- name: The name of the track.
- artist: The name of the artist or artists who performed the track.
- spotify preview url: A URL to a 30-second preview of the track on Spotify.
- spotify id: The Spotify ID for the track, used to uniquely identify it on the Spotify platform.
- tags: Tags associated with the track, often used for categorization or additional metadata.
- genre: The genre to which the track belongs.
- year: The year the track was released.
- duration ms: The length of the track in milliseconds.
- danceability: A measure of how suitable a track is for dancing, based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. Values range from 0.0 to 1.0.
- energy: A measure from 0.0 to 1.0 representing a perceptual measure of intensity and activity. Energetic tracks feel fast, loud, and noisy.
- key: The key in which the track is composed, represented as an integer. Integers map to pitches using standard Pitch Class notation.
- loudness: The overall loudness of the track in decibels (dB). Loudness values are averaged across the entire track.
- mode: Indicates the modality (major or minor) of the track. Major is represented by 1 and minor is 0.
- speechiness: Detects the presence of spoken words in a track. Values range from 0.0 to 1.0, with higher values indicating more speech-like content.
- acousticness: A confidence measure from 0.0 to 1.0 of whether the track is acoustic.
- instrumentalness: Predicts whether a track contains no vocals. Values above 0.5 are intended to represent instrumental tracks.
- liveness: Detects the presence of an audience in the recording. Higher values represent an increased probability that the track was performed live.
- valence: A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Higher values sound more positive.
- tempo: The speed or pace of a given piece, measured in beats per minute (BPM).
- time signature: An estimated overall time signature of a track. The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure).

A. Dataset Correlation

The following figure shows the music info data correlation heatmap, a powerful tool for visualizing relationships between various features that define a song. The table rows and columns represent characteristics like tempo, danceability, or acousticness. The heatmap fills this table with color, with red indicating positive correlations (features that tend to go together) and blue signifying negative ones (features that often oppose each other). For instance, a red hue between "tempo" and "energy" suggests faster

songs often feel more energetic, while a blue patch between ”acousticness” and ”liveness” implies live recordings typically capture less acoustic purity. By deciphering these color-coded connections, developers can improve music recommendation systems by suggesting songs that align with a user's preferred musical fingerprint.

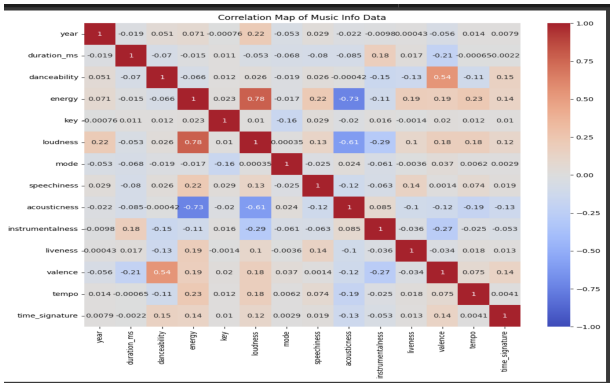

Fig. 1: Correlation Map of Music Info Data

B. *Exploratory Data Analysis (EDA)*

The exploratory data analysis involves several steps to uncover patterns and insights within the dataset:

- Descriptive Statistics: Summary statistics for numerical features, such as mean, median, and standard deviation, provide an overview of the data distribution.
- Missing Values: Identification and handling of missing values to ensure data quality. Visualization: Use of various plots (e.g., histograms, bar charts) to visualize the distribution and relationships between features.
- Correlation Analysis: Generation of a correlation heatmap to identify significant relationships between audio features.
- Temporal Analysis: Examination of trends over time, such as changes in song duration and popularity across different years and genres.

VI. Methods

A. *Fastai*

The fastai library was chosen for this project due to its high-level API that simplifies the implementation of deep learning models, including those for recommendation systems. The library offers prebuilt components for common tasks, such as data

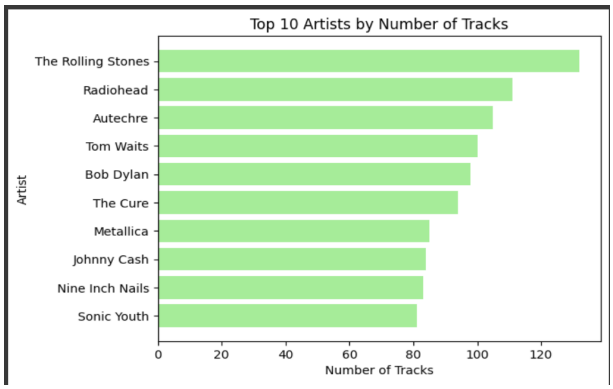

Fig. 2: Top 10 Artists Music Info Data

loading, model creation, and training and it provides a user-friendly interface that abstracts many of the complexities involved in deep learning.

CollabDataLoaders: This class is used to create data loaders specifically for collaborative filtering tasks. It helps in loading and preparing the data for training and validation.

collab learner: This function creates a learner object for collaborative filtering, which includes the model, optimizer, and data. It simplifies the process of training and evaluating collaborative filtering models.

lr find: This method is used to find an optimal learning rate for training the model. It runs a small range of learning rates across a few iterations of training and plots the loss vs. learning rate curve to help choose a suitable learning rate.

fit one cycle: This method trains the model using the 1cycle policy. It gradually increases the learning rate for the first half of training, then gradually decreases it for the second half. This approach often leads to faster training and better performance.

get preds: This method is used to get predictions from the model. It takes a DataLoader as input and returns predictions for the inputs in that DataLoader.

show batch: This method is used to visualize a batch of data from the DataLoader. It's often used to inspect the data before training the model.

BCEWithLogitsLossFlat: This is the loss function used for binary classification tasks with logits. It combines a sigmoid activation function with binary cross-entropy loss. In this case, it's used as the loss function for training the collaborative filtering model.

B. LSTM

A kind of recurrent neural network (RNN) architecture called Long Short-Term Memory (LSTM) was created to solve the vanishing gradient issue in conventional RNNs and enable them to efficiently learn long-term relationships in sequential input. Memory cells with different gates (such as input, forget, and output gates) that control the information flow inside the network make up Long Short-Term Memory Networks (LSTMs). Because of these gates, LSTMs may be trained to selectively retain or forget information over time. This makes them especially useful for modeling sequential data with long-range dependencies, such as text, human behavior, and time series. Using the provided code, LSTM is utilized to create a sequential recommendation model that predicts users' future preferences and suggests new items based on their past interactions with objects, in this example, songs. The model is trained on sequences of user-item interactions and then used to generate personalized recommendations for individual users based on their past behavior.

- LabelEncoder.fit transform: This method fits and transforms categorical variables into integer labels. It's used to encode user IDs and item IDs into numerical format for modeling.
- Create sequences: This function generates sequences of user-item interactions from the data. It's used to create training sequences for the LSTM model.
- SequenceDataset: This custom dataset class is used to create PyTorch datasets from sequences of user-item interactions. It's used to feed data into the model during training.
- LSTM Model: This class defines the architecture of the LSTM model. It specifies the embedding layer, LSTM layers, and fully connected layer for prediction.
- Train model: This function trains the LSTM model using the provided training data. It computes the loss and updates the model parameters using backpropagation.
- Recommend songs: This function generates song recommendations for a given user based on their past interactions. It uses the trained LSTM model to predict the user's preferences for unseen items.

C. PCA

Principal Component Analysis (PCA) is a dimensionality reduction technique that is frequently used to convert high-dimensional data into a lower-dimensional space while retaining as much of the original variance as feasible. Principal components, or the orthogonal axes that capture the most variance in the data, are found using PCA. The original data's patterns or directions of variation are represented by these major components. The code provided uses PCA to minimize the dimensionality of the interaction matrix between the user and the item. PCA facilitates tasks like recommendation and similarity analysis by projecting the original data into a lower-dimensional space defined by the principle components. This helps find latent elements or features that reflect users' preferences and items' properties.

- LabelEncoder.fit transform: This method fits and transforms categorical variables into integer labels. It's used to encode user IDs and item IDs into numerical format for modeling.
- PCA: Principal Component Analysis is carried out by this scikit learn class. By projecting the data onto a lower-dimensional subspace, it preserves the majority of the variance in the data while reducing its dimensionality.
- Cosine similarity: This function calculates a vector's cosine similarity. In the PCA reduced space, it is employed to quantify the similarity between user and item vectors.
- Get recommendations pca: This function uses the user's PCA transformed vector and the PCA-transformed item vectors to provide item suggestions for the specified user. The user vector and each item vector are compared using the cosine similarity calculation, and the top-N items with the highest similarity scores are chosen as suggestions.

VII. Results

Fastai Results: The collaborative filtering model's training demonstrated a consistent enhancement over a period of 15 epochs, as demonstrated by the decrease in training and validation losses and the rise in custom accuracy. The model's better fit on the data was first demonstrated by the training loss, which dropped from 0.3406 to 0.3076, and

the validation loss, which dropped from 0.3452 to 0.3172. Simultaneously, the custom accuracya measure of the percentage of accurate recommendationsexhibited a notable improvement, increasing from 0.5257 during the first epoch to 0.8015 by the end of the study. This trend demonstrates how, over the training period, the model's ability to produce appropriate music recommendations increased.

epoch	train_loss	valid_loss	custom_accuracy	time
0	0.340629	0.345212	0.525692	00:30
1	0.324094	0.332272	0.647484	00:30
2	0.332298	0.338870	0.586708	00:39
3	0.334141	0.341175	0.570672	00:38
4	0.332756	0.340009	0.583337	00:39
5	0.332461	0.339582	0.583434	00:37
6	0.330642	0.337402	0.600385	00:39
7	0.327907	0.335007	0.621478	00:38
8	0.323194	0.332614	0.644450	00:36
9	0.321341	0.328853	0.678594	00:38
10	0.316258	0.325057	0.719191	00:37
11	0.313674	0.321722	0.755069	00:37
12	0.309198	0.318954	0.782519	00:36
13	0.307541	0.317470	0.799856	00:36
14	0.307639	0.317158	0.801541	00:34

Fig. 3: Results of Training Epochs for Fastai Model

The least and most loved tracks as determined by the model are found through the analysis of track bias embeddings. "At Giza" by Om (2006), "Leaving Eden" by Antimatter (2007), and "Hooked On A Feeling" by B.J. Thomas (2011) are the songs that people dislike the most. On the other hand, the most well-liked tracks are "Halo" by Depeche Mode (1990), "Alejandro" by Lady Gaga (2010), and "Revelry" by Kings of Leon (2008). Based on the training data, these insights draw attention to the model's preferences and biases.

The test accuracy of the recommendations for user'4e11f45d732f4861772b2906f81a7d384552ad12' is 0.2667. The following figure shows a few recommended songs and Test Accuracy.

LSTM Analysis and Results: After 60 epochs of training, the LSTM model's loss and accuracy gradually improved, indicating the model's capacity for learning. The model was used to provide music recommendations for a particular user (user ID 7621) after training. The recommendation feature employed the trained LSTM model to estimate the next likely songs the user will like, based on sequence processing of the user's historical listening data. The model's capacity to offer tailored music recommendations based on the user's previous listening behavior was then demonstrated when the suggested songs were extracted and shown.

```
'Riot Van, Arctic Monkeys, 2006' 'Last Living Souls, Gorillaz, 2005'
 'South Is Only a Home, The Fiery Furnaces, 2003'
 'Kreuzberg, Bloc Party, 2007' 'Instant Street, dEUS, 1999'
 'Opus Eclipse, Therion, 1996' 'My List, The Killers, 2006'
 'N.Y., Doves, 2012' 'Cicadas, Deerhunter, 2008'
 'I Only Think of You, The Horrors, 2009'
 'Misty Mountains, Richard Armitage, 2012' 'Hands Open, Snow Patrol, 2009'
 'Sister Morphine, The Rolling Stones, 1971' 'Ragoo, Kings of Leon, 2007'
 'Simon, Lifehouse, 2000' 'Love Dog, TV on the Radio, 2008'
 'Tomorrow Comes Today, Gorillaz, 2011'
 'East St. Louis Toodle-Oo, Steely Dan, 1985'
 'Macadam Cowboy, Mando Diao, 2007' 'Money Power Respect, Diplo, 2014'
 'Zither, R.E.M., 2014' "You're So Damn Hot, OK Go, 2006"
 'English Civil War, The Clash, 2008' 'Them Eyes, The Black Keys, 2002'
 'Watch the Tapes, LCD Soundsystem, 2010']
Test Accuracy: 0.23333333432674408
```

Fig. 4: Recommended songs and Test Accuracy

```
Epoch 46/60, Loss: 1.2554402177532513, Accuracy: 0.675243089697367
Epoch 47/60, Loss: 1.2551593056155577, Accuracy: 0.6760624931716377
Epoch 48/60, Loss: 1.2016170426375337, Accuracy: 0.6883535452856987
Epoch 49/60, Loss: 1.4435868105954595, Accuracy: 0.6321970938490112
Epoch 50/60, Loss: 1.3314953438109822, Accuracy: 0.6546760624931717
Epoch 51/60, Loss: 1.2043177208138838, Accuracy: 0.6848027968971921
Epoch 52/60, Loss: 1.0847953448279037, Accuracy: 0.7106959466841473
Epoch 53/60, Loss: 0.9735064696934488, Accuracy: 0.7379820823773626
Epoch 54/60, Loss: 1.0441392262776692, Accuracy: 0.7190265486725663
Epoch 55/60, Loss: 1.0876507241692808, Accuracy: 0.7111056484212827
Epoch 56/60, Loss: 1.0161554546923273, Accuracy: 0.7255544630175899
Epoch 57/60, Loss: 0.9835537386437258, Accuracy: 0.7362340216322517
Epoch 58/60, Loss: 1.045376708938016, Accuracy: 0.7234786408827707
Epoch 59/60, Loss: 1.065818818493022, Accuracy: 0.7168960996394624
Epoch 60/60, Loss: 1.1933849338028166, Accuracy: 0.6921774281656288
```

Fig. 5: LSTM model training Loss and Accuracy

```
LSTMModel(
  (embedding): Embedding(15676, 50)
  (lstm_layers): ModuleList(
    (0): LSTM(50, 100, batch_first=True)
    (1): LSTM(100, 100, batch_first=True)
  )
  (fc): Linear(in_features=100, out_features=15676, bias=True)
)
```

Fig. 6: LSTM Model

PCA Analysis and Results: The code conducts Principal Component Analysis (PCA) on the interaction matrix derived from the listening history data. It prints the shapes of the user and item PCA matrices, showcasing the dimensionality reduction achieved by PCA. The explained variance ratio for each principal component is displayed, along with the cumulative explained variance.

Fig. 7: Recommended songs for user ID 7621

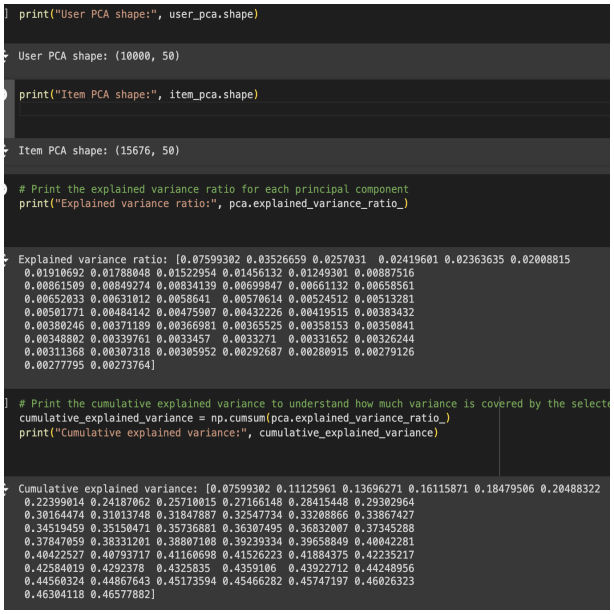

Fig. 8: PCA Shape and Variance

Recommendations Generation: Recommendations are generated for a specific user('4e11f45d732f4861772b2906f81a7d384552ad12') based on cosine similarity scores.

The top recommended songs, along with their names, artists, and release years, are listed.

When comparing the music recommendation models, Fastai stands out as the most attractive option due to its notably enhanced capacity to propose songs that are relevant. Fastai's proprietary accuracy metric, which gauges how accurately it can recommend music, increased dramatically from 0.5257 to 0.8015 throughout training. When compared to the other models, this is unique. All three models are capable of producing suggestions, but Fastai goes

Fig. 9: Recommended songs for the mentioned user

above and above by offering a user-specific test accuracy of 0.2667, which enables evaluation of how closely recommendations match user preferences.

Although the LSTM model does not have particular metrics to measure the progress in recommendation accuracy, it does show a gradual improvement in loss and accuracy during training. However, PCA doesn't specifically address how successful the recommendations are; instead, it concentrates on dimensionality reduction, which is an important preprocessing step for recommendation systems. In conclusion, Fastai is the best model for more research due to its remarkable increase in bespoke accuracy and user-centric approach to suggestions.

VIII. Conclusion

In this project, we developed an advanced music recommender system leveraging the Fastai library, with a focus on integrating Long Short-Term Memory (LSTM) networks and Principal Component Analysis (PCA) for enhanced performance. The Fastai library provided a robust framework that streamlined the implementation of our models, offering high-level APIs and tools for efficient model training and fine-tuning. By utilizing LSTMs, we were able to effectively capture the sequential nature of user listening patterns, allowing the system to make more accurate and personalized music recommendations. This approach addressed the limitations of traditional methods by adapting to the evolving preferences of users and considering the temporal dynamics of their listening behavior. Additionally, PCA was employed to reduce the dimensionality of the dataset, which significantly improved the system's scalability and computational efficiency. By transforming the data into a

set of principal components, we retained the essential patterns and structures within the user data while simplifying the computational requirements. This was crucial for managing the vast amounts of information generated by millions of users and ensuring real-time processing capabilities. The combination of LSTM networks for capturing sequential behavior and PCA for feature reduction, facilitated by Fastai, resulted in a more dynamic, scalable, and personalized music recommender system. This project not only demonstrates the potential of advanced machine learning techniques in enhancing music streaming services but also sets a foundation for future innovations in recommendation systems.

REFERENCES

[1] Lakey, Daniel and Schlippe, Tim. (2024) *A Comparison of Deep Learning Architectures for Spacecraft Anomaly Detection*

[2] Joshi, Saurav and Jain, Tanuj and Nair, Nidhi. (2021) *Emotion based music recommendation system using LSTM-CNN architecture*

[3] Yadav, Vikash and Shukla, Rati and Tripathi, Aprna and Maurya, Anamika and others. (2021) *A new approach for movie recommender system using K-means Clustering and PCA*

[4] Chen, Hung-Chen and Chen, Arbee LP. (2001) *A music recommendation system based on music data grouping and user interests*

[5] Amiri, Babak and Shahverdi, Nikan and Haddadi, Amirali and Ghahremani, Yalda. (2024) *Beyond the Trends: Evolution and Future Directions in Music Recommender Systems Research*

[6] Shigetomi, Ryunosuke and Nishida, Hiroko and Sawai, Kenichi and Ushiama, Taketoshi. (2024) *Integrating Repeat Listening Patterns for Enhanced Music Recommendation*

[7] Panchireddy, Swathi and Yanda, Nagamani. (2024) *Personalized Music Recommendation through Machine Learning*

[8] Panchireddy, Swathi and Yanda, Nagamani. (2024) *Personalized Music Recommendation through Machine Learning*