Music Recommendation System

Aarushi Agarwal, Athena Hartigan
Wake Forest University
1834 Wake Forest Rd
agara21@wfu.edu, hartag21@wfu.edu

Abstract

With the rapid expansion of digital music libraries and streaming platforms, providing personalized and relevant music recommendations has become essential for enhancing user engagement and satisfaction. This project aims to develop a music recommendation system that uses content-based filtering to offer accurate, diverse, and personalized song suggestions. By leveraging advanced audio feature extraction from tracks, the system seeks to address challenges such as the cold-start problem, data sparsity, and the balance between familiar and novel recommendations. In this project, we collect and process song metadata, and audio features to train various machine learning models, including similarity-based methods for content filtering. Additionally, we extract musical characteristics such as tempo, key, and spectral features from audio files to enrich the recommendation process.

Demo: https://music-recommender-svm.streamlit.app/

Github link: https://github.com/athenahartigan/music-recommender

1 Introduction

1.1 Background

Music recommendation systems have become a fundamental part of digital music streaming platforms, significantly transforming how people discover and listen to music. Modern music recommendation systems have evolved to leverage machine learning and artificial intelligence techniques to provide personalized listening experiences. These systems predominantly rely on two main approaches: collaborative filtering and content-based filtering.

Collaborative filtering focuses on analyzing user interactions and finding patterns in their behavior to suggest tracks that similar users have enjoyed. This technique has proven effective in uncovering user preferences without needing explicit information about the songs themselves. On the other hand, content-based filtering utilizes the features of the songs, such as tempo, key, genre, lyrics, and rhythm, to recommend tracks that share similar characteristics to what a user has already liked.

In this project, by extracting and analyzing deeper musical characteristics directly from audio tracks, we intend to enhance the personalization of recommendations and improve user satisfaction in music exploration.

1.2 Motivation

The abundance of available music can make it hard to find new favorite songs. Most music recommendation systems recommend songs based on what similar people listened to. This can prevent users from finding new music that might not be as popular. We plan to solve this issue by creating a recommendation system based on the music itself. Those who tend to like certain songs should like ones similar to them.

Another thing is that our system would be able to recommend music that is very different from what our user is used to. This recommendation system would not only prevent the monopoly of certain artists or songs being popular but would also allow users to expand their horizon as they can explore genres that they may have never considered before. Our system in conclusion is focusing solely on the user and their music, creating a more personalized recommendation than current systems. By capturing nuances in music features, we are able to recommend songs that might otherwise be missed by the user.

1.3 Challenges

Cold-Start Problem: The model could have difficulty recommending items to new users with no history or new items with no user interactions. While collaborative filtering excels at suggesting popular tracks, it may not be as effective at introducing users to lesser-known songs or genres they might enjoy. Content-based systems, while better at personalizing, can sometimes limit the variety of recommendations, leading to repetitive listening experiences. If the audio features do not have a direct correlation with the genres and similarities of the songs, it would be hard to assess whether it is worth using them as features.

Scalability: Music streaming platforms handle massive amounts of data from millions of users and songs, requiring recommendation algorithms to be highly scalable. Ensuring that the system can make real-time recommendations for a growing user base without compromising performance is a technical challenge. It would also be a challenge to provide recommendations for songs that do not currently exist in the database. We would need to incorporate methods to extract audio features real time and make predictions based on that, which might be a long process.

2 Related work

Numerous music recommendation systems have been proposed, often relying heavily on user interaction data. While these systems can provide personalized recommendations for established users, they face limitations such as the cold-start problem and the challenge of recommending diverse and unexpected music.

To address these limitations, researchers have explored various techniques, including content-based filtering, collaborative filtering, and hybrid approaches. Content-based filtering recommends items based on their similarity in features, while collaborative filtering leverages user-item interactions. Hybrid approaches combine these techniques to improve recommendation accuracy. Deep learning models, such as neural networks and transformers, have been employed to capture complex patterns in user behavior and music features.

Additionally, research by Budhrani et al. [2], Elbir and Aydin [3], and Adiyansyah et al. [5] has demonstrated the effectiveness of music embeddings in capturing the inherent characteristics of songs. Genre classification techniques, as explored in [4], [7], and [12], can improve recommendation accuracy. Hybrid approaches, as suggested in [8] and [9], can leverage the strengths of both content-based and collaborative filtering. To address the challenges of cold start and data sparsity, techniques like personalized exploration strategies and diversification algorithms, as discussed in [6], can be employed.

By incorporating these techniques and addressing the limitations of existing approaches, we aim to develop a robust and effective music recommendation system that provides personalized and engaging listening experiences.

3 Methodology

Proposed framework

Data Collection: We plan to collect raw audio tracks from existing YouTube playlists that have been organized by genre, utilizing web scraping technologies such as the yt-dlp python library. We will visualize a subset of the data features to understand how the audio features are related to each other, and how they look in different genres. We will conduct a principal component analysis, plot frequency vs time, amplitude vs time graphs. We will extract the features such as tempo, harmony,

etc. from each audio track, storing it in a separate file using the python library Librosa, which extracts such features from raw tracks. The genre labels are encoded using LabelEncoder and the features are normalized using StandardScaler.

Predictions: We will use the extracted audio feature data to train machine learning models and test the accuracy of genre predictions. This will help us investigate how the audio features correspond to the genres. We will train and test multiple machine learning models and evaluate their performance. For example for the neural network, we use a linear activation function with mse as the loss function and Adam optimizer. We use randomized grid search to find the best model hyperparameters to train it on the features of the songs in the dataset. Our overall workflow is: when a user inputs a song, the model predicts the features of the song using a trained model, calculates the similarity indices of the songs in the dataset using cosine similarity to the inputted song, and recommends the most similar songs.

The proposed music recommendation system leverages a combination of techniques to provide recommendations:

Training Input (for the Neural Network Model)

- The input to the model consists of the numerical features of songs extracted from the dataset (e.g., tempo, rhythm, loudness, etc.).
- **Shape:** $(num\_samples, num\_features)$, where $num\_features$ is the number of numeric audio features in the dataset.
- **Standardized values:** All features are scaled using `StandardScaler` to have mean 0 and variance 1.

Recommendation Input

- A single song name (e.g., `'Radiohead - Creep.wav'`) provided by the user.
- The numeric feature vector corresponding to the provided song, extracted and scaled using the `StandardScaler`.

Model Output (Neural Network Predictions)

- The model predicts a feature vector of the same shape as the input vector.
- **Shape:** $(1, num\_features)$, where each feature represents the model's prediction of the corresponding feature values for the input song.

Recommendation Output

- A list of recommended songs and their similarity scores:
  - **Recommended Songs:** Names of the most similar songs based on cosine similarity.
  - **Similarity Scores:** Cosine similarity values for the top recommendations.

4 Exploration

4.1 Implementation and Baseline Comparison

Genre Prediction: To optimize the performance of our machine learning models for predicting song genres, we employed GridSearchCV to meticulously tune hyperparameters. This involved defining a grid of potential values for each hyperparameter and systematically evaluating their combinations using 5-fold cross-validation. The hyperparameter combination that yielded the lowest mean squared error (MSE) was selected as the optimal configuration for each model.

Key Hyperparameters and Insights:

No hyperparameters were tuned for Naive Bayes

Table 1: Hyperparameters for Stochastic Gradient Descent

Hyperparameter	Value 1	Value 2
max_iter	1000	5000
alpha	0.0001	0.001
penalty	l2	l1
penalty	elasticnet	

Table 2: Hyperparameters for K-Nearest Neighbors

Hyperparameter	Value 1	Value 2	Value 3	Value 4
n_neighbors	5	10	15	19
weights	uniform	distance		
algorithm	auto	ball_tree	kd_tree	brute

Table 3: Hyperparameters for Decision Tree

Hyperparameter	Value 1	Value 2	Value 3
max_depth	10	20	30
min_samples_split	2	10	20
min_samples_leaf	1	5	10

Table 4: Hyperparameters for Random Forest

Hyperparameter	Value 1	Value 2	Value 3
n_estimators	100	500	1000
max_depth	None	10	20
min_samples_split	2	5	10
min_samples_leaf	1	2	4

Table 5: Hyperparameters for Support Vector Machine

Hyperparameter	Value 1	Value 2	Value 3
C	1	10	100
gamma	scale	auto	
kernel	linear	rbf	poly
decision_function_shape	ovo	ovr	

Table 6: Hyperparameters for Logistic Regression

Hyperparameter	Value 1	Value 2
solver	lbfgs	liblinear
C	0.1	1
C	10	

Table 7: Hyperparameters for Neural Network

Hyperparameter	Value 1	Value 2
solver	lbfgs	adam
alpha	1e-5	1e-4
alpha	1e-3	
hidden_layer_sizes	(100,)	(5000, 10)

Table 8: Hyperparameters for Cross Gradient Booster

Hyperparameter	Value 1	Value 2
n_estimators	100	500
learning_rate	0.01	0.05
learning_rate	0.1	

Table 9: Hyperparameters for Cross Gradient Booster (Random Forest)

Hyperparameter	Value 1	Value 2
n_estimators	100	500
max_depth	3	6
max_depth	9	
min_child_weight	1	3
min_child_weight	5	

The model with the best performance was selected as the final recommendation:

No hyperparameters were tuned for Naive Bayes

Table 10: Chosen Hyperparameters for Stochastic Gradient Descent

Hyperparameter	Value
alpha	0.01
max_iter	1000
penalty	l2

Table 11: Chosen Hyperparameters for K-Nearest Neighbors

Hyperparameter	Value
algorithm	auto
n_neighbors	10
weights	uniform

Table 12: Chosen Hyperparameters for Decision Trees

Hyperparameter	Value
max_depth	30
min_samples_leaf	5
min_samples_split	20

Table 13: Chosen Hyperparameters for Random Forest

Hyperparameter	Value
hline max_depth	None
min_samples_leaf	1
min_samples_split	2
n_estimators	100

Table 14: Chosen Hyperparameters for Support Vector Machine

Hyperparameter	Value
C	10
decision_function_shape	ovo
gamma	scale
kernel	linear

Table 15: Chosen Hyperparameters for Logistic Regression

Hyperparameter	Value
C	10
solver	liblinear

Table 16: Chosen Hyperparameters for Neural Networks

Hyperparameter	Value
alpha	0.001
hidden_layer_sizes	(100,)
solver	adam

Table 17: Chosen Hyperparameters for Cross Gradient Booster

Hyperparameter	Value
learning_rate	0.1
n_estimators	100

Table 18: Chosen Hyperparameters for Cross Gradient Booster (Random Forest)

Hyperparameter	Value
max_depth	9
min_child_weight	1
n_estimators	500

The MSE, MAE, and R^2 is as follows:

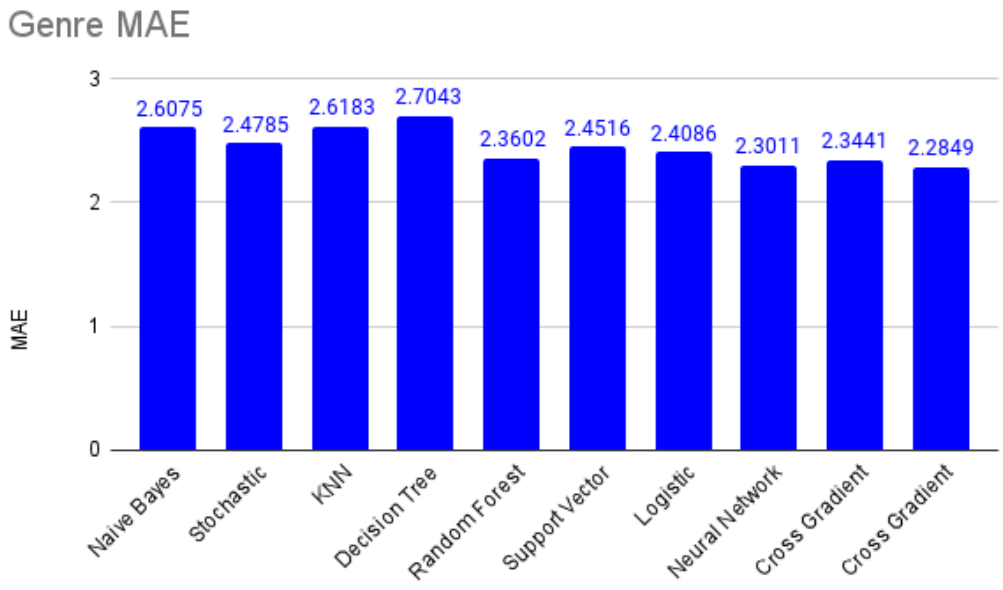

Figure 1: Genre MAE

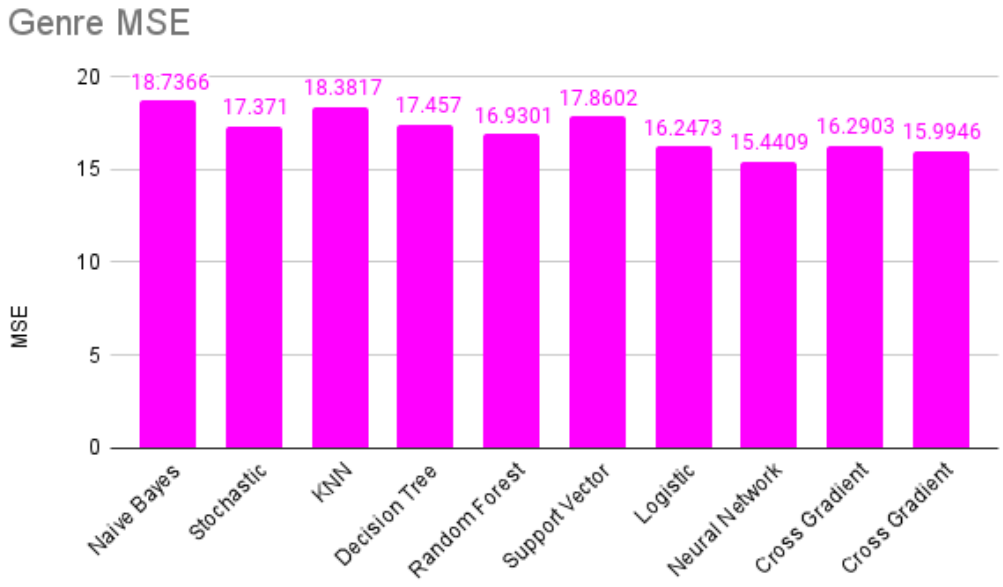

Figure 2: Genre MSE

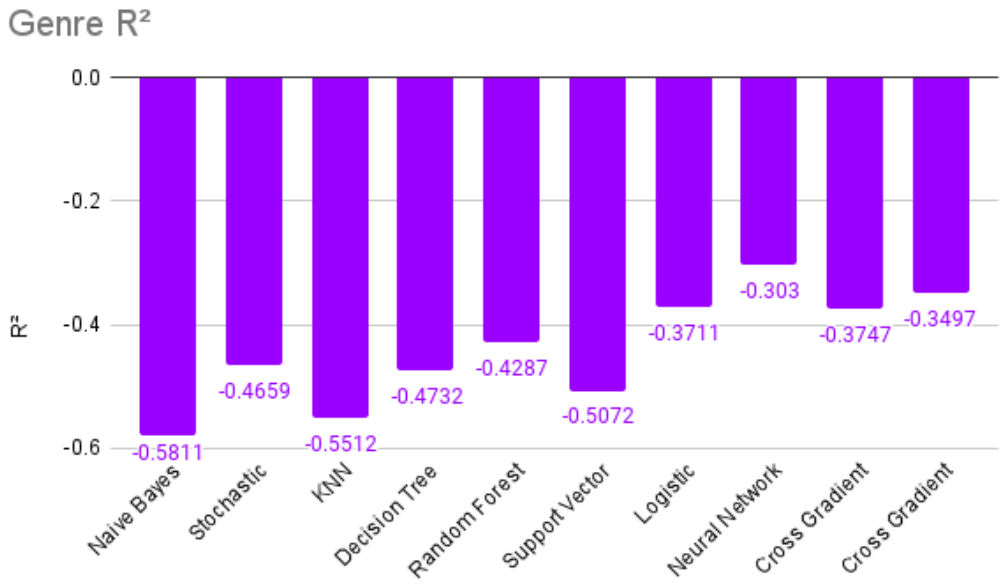

Figure 3: Genre R²

To gain deeper insights into the underlying structure of song genres, we employed a combination of clustering and dimensionality reduction techniques. K-means clustering was utilized to partition the dataset into 13 clusters, corresponding to the predefined number of genres. This algorithm iteratively assigns data points to clusters and updates cluster centroids to minimize the within-cluster sum of squares. To optimize the clustering process, we set the number of clusters to 13 (n_clusters=13), limited the maximum number of iterations to 300 (max_iter=300), and ran the algorithm with 10 different initializations (n_init=10) to improve convergence.

For a more flexible approach, Fuzzy C-Means (FCM) clustering was employed, allowing data points to belong to multiple clusters with varying degrees of membership. This is particularly useful for genres that blend multiple styles. The key hyperparameters for FCM were set as follows: n_clusters=13, m=2, max_iter=150, and error=0.005.

To reduce the dimensionality of the data while preserving its essential characteristics, Principal Component Analysis (PCA) was applied. This linear dimensionality reduction technique identifies the principal components, which are linear combinations of the original features that capture the most variance in the data. The number of principal components to retain was determined using the elbow method or explained variance ratio.

To further explore the complex structure of the music dataset, t-Distributed Stochastic Neighbor Embedding (t-SNE) was utilized. This non-linear dimensionality reduction technique is particularly effective for visualizing high-dimensional datasets, such as those in music. Key parameters for t-SNE included: n_components=2, perplexity=30, learning_rate=200, and n_iter=1000.

Finally, an autoencoder was employed to learn a compressed representation of the input data. By training the autoencoder to reconstruct the original data from a lower-dimensional latent representation, we were able to extract the most salient features of the data. Key parameters for the autoencoder included: encoding_dim, epochs, and batch_size.

Despite some limitations in genre prediction accuracy, carefully selecting and tuning hyperparameters allowed us to gain valuable insights into the underlying structure and relationships between different music genres.

Our Results:

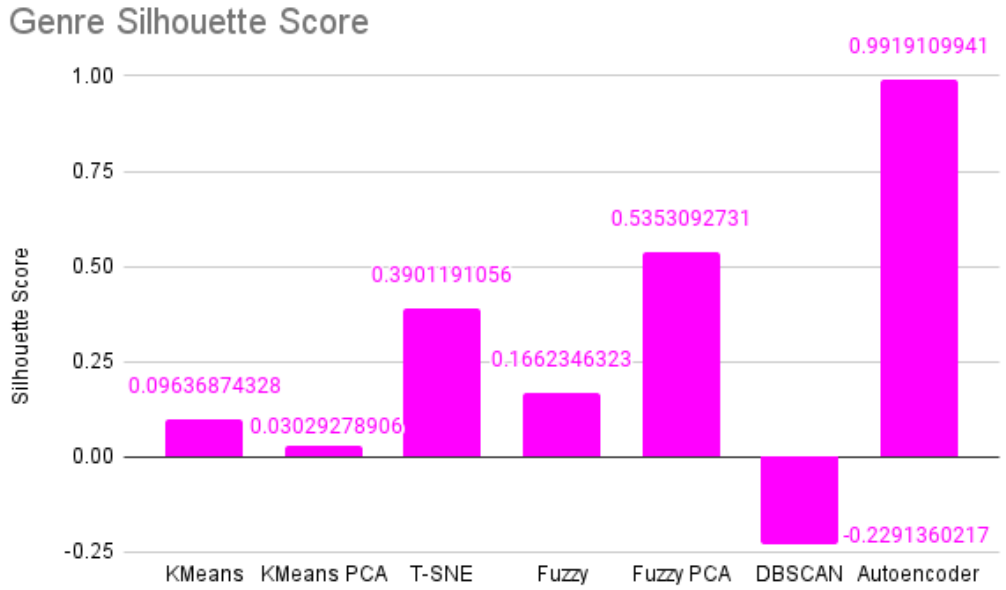

Figure 4: Genre Silhouette Score

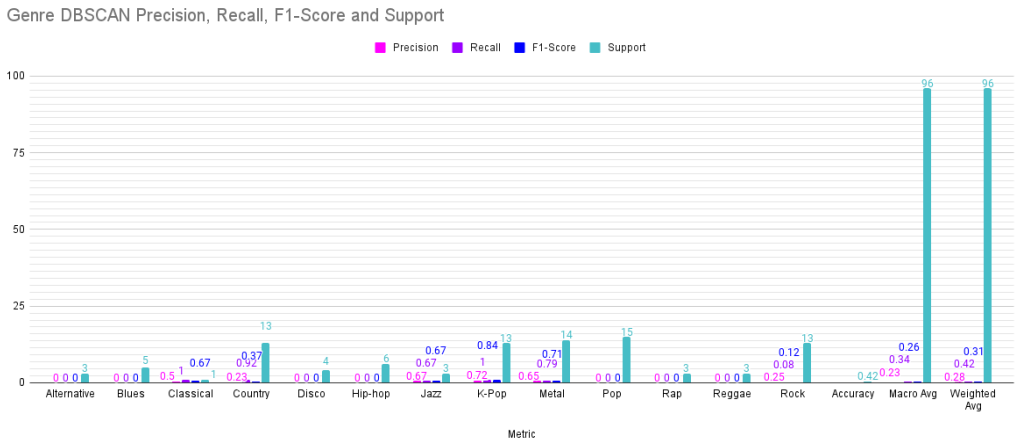

Figure 5: Genre DBSCAN Precision, Recall, F1-Score and Support

By combining supervised and unsupervised learning techniques, we were able to effectively predict song genres and gain insights into the underlying structure of the dataset. Further exploration and refinement of these techniques can lead to even more accurate and informative models.

Recommending Songs: To create a robust music recommendation system, we explored a variety of machine learning models, including Neural Networks, K-Nearest Neighbors (KNN), Decision Trees, Random Forest, and Support Vector Machines (SVM). For each model, we rigorously tuned hyperparameters to optimize performance.

For each model, we employed grid search or randomized search cross-validation to explore a wide range of hyperparameter combinations. The specific hyperparameters tuned for each model are as follows:

Table 19: Hyperparameters for Neural Network

Hyperparameter	Value 1	Value 2	Value 3
Optimizer	Adam	RMSprop	
Kernel Initializer	Uniform	Normal	
Number of Neurons (Hidden Layers)	64	128	256
Batch Size	16	32	64
Number of Epochs	50	100	150

Table 20: Hyperparameters for K-Nearest Neighbors

Hyperparameter	Value 1	Value 2	Value 3	Value 4	Value 5
Number of Neighbors	3	5	7	9	11
Weights	Uniform	Distance-based			
Algorithm	Auto	Ball Tree	KD Tree	Brute Force	

Table 21: Hyperparameters for Decision Tree

Hyperparameter	Value 1	Value 2	Value 3	Value 4	Value 5	Value 6
max_depth	None	10	20	30	40	50
min_samples_split	2	5	10	20		
min_samples_leaf	1	2	4	8		

Table 22: Hyperparameters for Random Forest

Hyperparameter	Value 1	Value 2	Value 3	Value 4
n_estimators	50	100	200	
max_depth	None	10	20	30
min_samples_split	2	5	10	
min_samples_leaf	1	2	4	
bootstrap	True	False		

Table 23: Hyperparameters for Support Vector Machine

Hyperparameter	Value 1	Value 2	Value 3	Value 4
C	0.1	1	10	100
gamma	scale	auto		
kernel	linear	poly	rbf	sigmoid

After tuning hyperparameters, we trained and evaluated each model on a validation set using metrics like Mean Squared Error (MSE), Mean Absolute Error (MAE), and R-squared (R^2). To obtain more robust performance estimates, we also performed 5-fold cross-validation for each model.

The model with the best performance on the validation set was selected as the final recommendation model as follows:

Table 24: Chosen Hyperparameters for Neural Network

Hyperparameter	Value
Optimizer	Adam
Kernel Initializer	Normal
Number of Neurons (Hidden Layers)	64, 64, 64
Batch Size	16
Number of Epochs	150

Table 25: Chosen Hyperparameters for KNN

Hyperparameter	Value
Number of Neighbors	7
Weights	Distance-based
Algorithm	Auto

Table 26: Chosen Hyperparameters for Decision Tree

Hyperparameter	Value
max_depth	20
min_samples_split	20
min_samples_leaf	8

Table 27: Chosen Hyperparameters for Random Forest

Hyperparameter	Value
n_estimators	200
max_depth	None
min_samples_split	2
min_samples_leaf	1
bootstrap	True

Table 28: Chosen Hyperparameters for SVM

Hyperparameter	Value
C	100
gamma	scale
kernel	linear

The MSE, MAE, and R^2 is as follows:

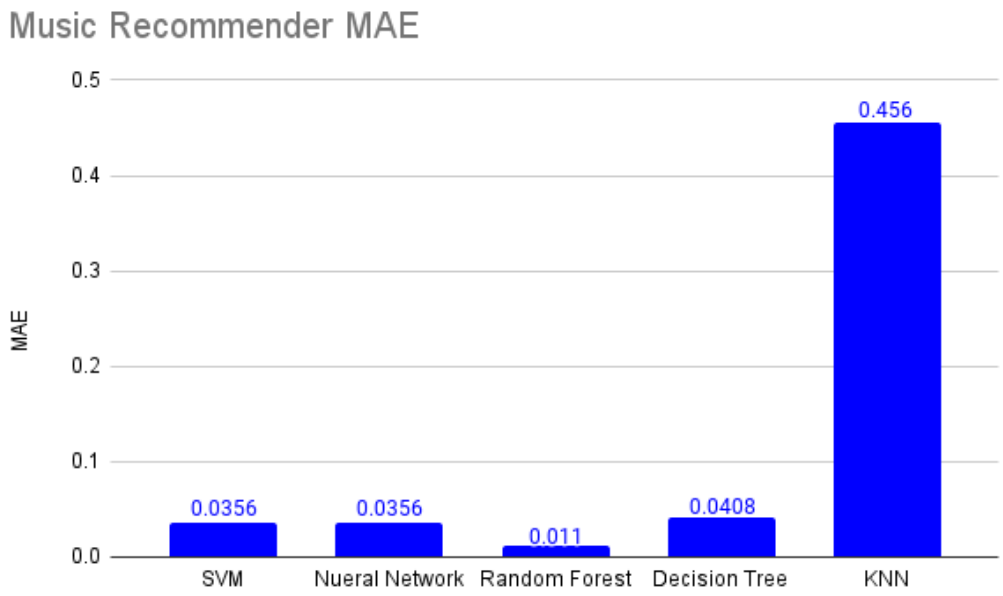

Figure 6: Music Recommender MAE

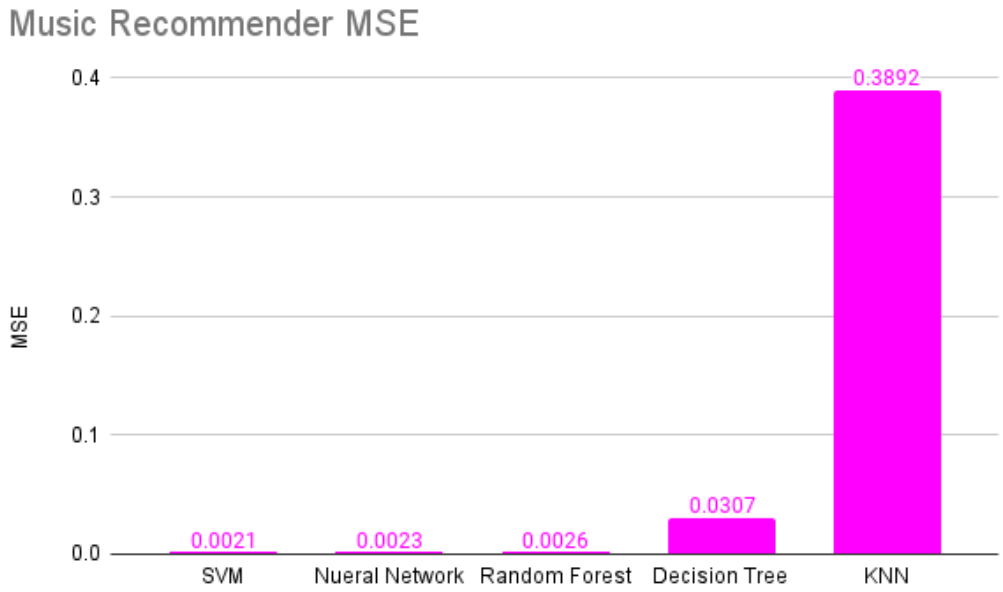

Figure 7: Music Recommender MSE

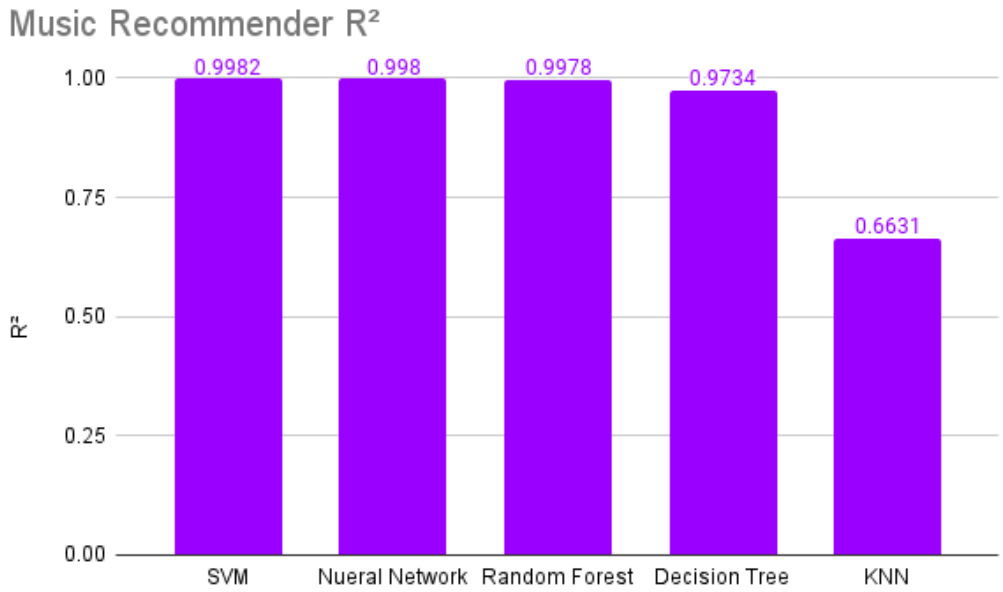

Figure 8: Music Recommender R^2

Based on the performance metrics, the Support Vector Machine (SVM) model with the specified hyperparameters emerged as the top-performing model. It achieved the lowest MSE and MAE, along with the highest R^2, indicating superior predictive accuracy and a strong fit to the data.

The SVM model, combined with cosine similarity for finding similar songs, was selected as the final recommendation system. This approach effectively leverages the strengths of SVM's powerful predictive capabilities and cosine similarity's ability to measure feature similarity.

For a given query song, the system:

Feature Extraction: Extracts relevant features from the song, such as tempo, rhythm, and harmonic content.

Model Prediction: Uses the trained SVM model to predict similar songs based on the extracted features.

Similarity Calculation: Calculates the cosine similarity between the query song's feature vector and the feature vectors of all other songs in the dataset. Cosine similarity measures the cosine of the angle between two vectors, providing a measure of their similarity in high-dimensional space.

Recommendation: Ranks the songs based on their cosine similarity to the query song and recommends the 5 top-ranked songs to the user.

5 Project Summary

Conclusion:

This project delves into the construction of a robust music recommendation system, leveraging a diverse array of machine learning models and clustering techniques.

Genre Recommendation:

While clustering techniques like K-Means, PCA, and t-SNE were employed to identify genre clusters, accurate genre prediction proved to be a challenging task. The inherent ambiguity and overlap between genres often limited the effectiveness of automated classification methods.

Music Recommendation:

To address the core task of music recommendation, a variety of machine learning models were implemented and evaluated based on metrics such as Mean Squared Error (MSE), Mean Absolute Error (MAE), and R-squared.

Neural Networks: Demonstrated exceptional performance, with low MSE and MAE, and a high R-squared value.

K-Nearest Neighbors (KNN): While effective, KNN generally exhibited higher error rates compared to more advanced models.

Decision Trees: Showed promising results, particularly in terms of interpretability.

Random Forest: Outperformed individual decision trees by leveraging ensemble learning techniques.

Support Vector Machines (SVM): Achieved competitive performance, especially in high-dimensional spaces.

Based on these performance metrics, Neural Networks, Random Forest, and SVM emerged as the top-performing models. They showcased their ability to capture complex patterns within the data, leading to accurate recommendations.

While this project provides a strong foundation for music recommendation systems, future work could focus on further refining hyperparameters, exploring advanced feature engineering, and potentially incorporating more sophisticated techniques like deep learning to enhance model performance and user experience. Additionally, addressing the challenges associated with accurate genre prediction remains an important area for future research.

Workload Distribution:

Aarushi: Data Collection, Feature Extraction/Engineering, Exploratory Data Analysis, Model Training, Streamlit Interface

Athena: Data Collection, Model Training, Fine-Tuning, Hyperparameter Optimization

6 References

[1] https://github.com/athenahartigan/music-recommender

[2] A. Budhrani, A. Patel and S. Ribadiya, "Music2Vec: Music Genre Classification and Recommendation System," 2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 2020, pp. 1406-1411, doi: 10.1109/ICECA49313.2020.9297559.

[3] Elbir, A. and Aydin, N. (2020), Music genre classification and music recommendation by using deep learning. Electron. Lett., 56: 627-629. https://doi.org/10.1049/el.2019.4202

[4] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," in IEEE Transactions on Speech and Audio Processing, vol. 10, no. 5, pp. 293-302, July 2002, doi: 10.1109/TSA.2002.800560.

[5] Adiyansyah, A. A. S. Gunawan, D., & Suhartono, H. (2019). Music recommender system based on genre using convolutional recurrent neural networks. Procedia Computer Science, 157, 99–109. https://doi.org/10.1016/j.procs.2019.08.146

[6] Yu Liang and Martijn C. Willemsen. 2019. Personalized Recommendations for Music Genre Exploration. In Proceedings of the 27th ACM Conference on User Modeling, Adaptation and Personalization (UMAP '19). Association for Computing Machinery, New York, NY, USA, 276–284. https://doi.org/10.1145/3320435.3320455

[7] J. Singh and V. K. Bohat, "Neural Network Model for Recommending Music Based on Music Genres," 2021 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 2021, pp. 1-6, doi: 10.1109/ICCCI50826.2021.9402621.

[8] Xuan Zhu, Yuan-Yuan Shi, Hyoung-Gook Kim and Ki-Wan Eom, "An integrated music recommendation system," in IEEE Transactions on Consumer Electronics, vol. 52, no. 3, pp. 917-925, Aug. 2006, doi: 10.1109/TCE.2006.1706489.

[9] Lampropoulos, A. S., Lampropoulou, P. S., & Tsihrintzis, G. A. (2012). A Cascade-Hybrid Music Recommender System for mobile services based on musical genre classification and personality

diagnosis. Multimedia Tools and Applications, 59(1), 241-258. https://doi.org/10.1007/s11042-011-0742-0.

[10] Schedl, Markus, Hamed Zamani, Ching-Wei Chen, Yashar Deldjoo, and Mehdi Elahi. "Current challenges and visions in music recommender systems research." International Journal of Multimedia Information. Retrieval 7, no. 2 (2018): 95-116. https://doi.org/10.1007/s13735-018-0154-2.

[11] Paul, Dip & Kundu, Subhradeep. (2020). A Survey of Music Recommendation Systems with a Proposed Music Recommendation System, doi: 10.1007/978-981-13-7403-6_26.

[12] Kilickaya, O. (2024). Genre classification and musical features analysis. International Journal of Latest Engineering Research and Applications (IJLERA), 9(4), 18-33. Retrieved from https://www.researchgate.net/publication/379958695_Genre_Classification_and_Musical_Features_Analysis