ISSN: 2584-0495 Vol. 3, Issue 8, pp. 1715-1723

International Journal of Microsystems and IoT

ISSN: (Online) Journal homepage: https://www.ijmit.org

AI Meets Music: Machine Learning for Indian Classical Raag Classification

Shreya S. Aigalikar, Anuradha C. Phadke, Jyoti Lele

Cite as: Aigalikar, S. S., Phadke, A. C., & Lele, J. (2025). AI Meets Music: Machine Learning for Indian Classical Raag Classification. International Journal of Microsystems and IoT, 3(8), 1715–1723. https://doi.org/10.5281/zenodo.18172527

Published online: 05 August 2025

. https://doi.org/10.5281/zenodo.18172527

International Journal of Microsystems and IoT
Vol.3, Issue 8, pp.1715-1723; DOI: . https://doi.org/10.5281/zenodo.18172527

AI Meets Music: Machine Learning for Indian Classical Raag Classification

Shreya S. Aigalikar, Anuradha C. Phadke, Jyoti Lele

School of Electronics and Communication, Dr. Vishwanath Karad MIT World Peace University, Pune, India

ABSTRACT
Hindustani Classical Music is built upon intricate melodic structures known as Raags, each defined by unique tonal, temporal, and stylistic elements. Their recognition requires nuanced understanding and years of training, making automated classification a challenging task. This paper presents a machine learning framework for identifying Raag Yaman and Raag Malkauns, both standardized to A# pitch for consistency. Primary audio samples were recorded in a controlled studio environment, pre-processed to extract frequency-based statistical features (mean, median, and standard deviation), and used to train a Random Forest classifier. The proposed model achieved 91% accuracy in differentiating between the two Raags, demonstrating that simple statistical features, when paired with an ensemble learning approach, can yield high performance. The study contributes a reproducible, data-driven method for computational Raag analysis and highlights its potential in music education, digital archiving, and AI-assisted performance evaluation.

KEYWORDS
Hindustani Classical Music, Machine Learning, Audio Processing, Random Forest Classifier

1. INTRODUCTION

Hindustani Classical Music is an extensive and deeply codified tradition where improvisation and structure coexist within the framework of Raags. A Raag prescribes an ascending (Aroh) and descending (Avroh) sequence of notes, along with ornamentations, characteristic phrases (Pakad), and a specific emotional or temporal association. Each performance balances strict adherence to these rules with interpretive freedom, producing variations that challenge even seasoned listeners in precise identification. Traditional Raag classification relies on expert musicianship, using methods such as That-Raag grouping based on primary notes, Raag-Ragini classification based on distinguishing features, and Shuddha–Chayalag– Sankeerna grouping for Raags derived from multiple sources. While effective in pedagogy, such approaches are inherently subjective and time-consuming.The rise of computational musicology and machine learning has opened new opportunities for automated Raag identification. By leveraging supervised learning algorithms, large-scale audio datasets can be analyzed systematically, enabling objective classification based on measurable acoustic features. Among these algorithms, the Random Forest classifier is particularly attractive for structured, high-dimensional data due to its robustness, interpretability, and resistance to overfitting.In this study, we focus on frequency-domain statistical descriptors extracted from high-quality studio recordings of Raag Yaman and Raag Malkauns, both tuned to A# for pitch uniformity. The objective is to demonstrate that even with minimal feature complexity, accurate classification can be achieved using ensemble learning.

This work forms a foundation for expanding computational tools in Indian classical music research and education. responses at various scales. This section discusses important methodologies, their application, and how they have deepened our understanding of cellular and tissue dynamics.

2. LITERATURE REVIEW:

Automated music classification has been studied extensively via learning algorithms plus feature types along with genres. Support Vector Machines were used in early methods to recognize genres showing they handled complex musical patterns well. Xu et al. used SVMs for multi-layer architectures, and their genre categorization outperformed distance-based classifiers. Deep learning models including Convolutional Neural Networks (CNNs) along with Recurrent Neural Networks (RNNs) often exceed customary algorithms whenever trained on sufficiently large as well as diverse datasets especially for audio clips of long duration as revealed by comparative analyses such as those by Ndou et al. Survey studies, such as ones including Scaringella et al., have catalogued techniques that are for music information retrieval. Feature extraction methods ranging from spectral descriptors up to temporal dynamics are important, as those are the studies that highlight it. Random Forest classifiers have emerged as competitive alternatives because deep learning offers strong performance, especially for tasks with limited training data or high-dimensional structured inputs. Parmar et al. stressed that they implemented it easily and it was resilient to parameter tuning, while Chaudhary et al. demonstrated that they improved multi-class accuracy upon optimizing features and selectively filtering data.

Audio analysis Random Forests exist outside music genre uses. These applications are used in bird call identification, speech processing, and environmental sound recognition which are specialized domains.. For example, Lele et al. converted bird audio to chromogram images for Vision Transformer-based classification, achieving 98% accuracy — an approach that inspired aspects of this work. Studies comparing SVM and Random Forest for audio classification (e.g., Ansari et al.) have shown performance variability depending on the nature of the acoustic features and the diversity of the dataset. In the context of music, feature choice significantly influences classification accuracy. Combining MIDI and raw audio features, as explored by Cataltepe et al., yields higher precision than using either alone, underscoring the role of feature engineering. Deep learning models have also been applied to music recommendation systems (Elbir and Aydin), demonstrating the versatility of AI in personalized content delivery. Building on these findings, our study investigates a Random Forest-based approach using minimal but discriminative statistical features from frequency data to classify two distinct Hindustani Raags. This focus on simplicity and reproducibility distinguishes our work from feature-heavy or deep-learning-dominant strategies, while still achieving competitive accuracy. Automated classification of Indian classical Raags remains a challenging task due to the intricacies and variations in their melodic structures. The objective of this study is to develop a machine learning-based system capable of accurately identifying Raags using extracted frequency parameters from audio data. This work aims to bridge the gap between traditional musicology and computational techniques by providing a reliable and data-driven approach for Raag classification.

3. METHODOLOGY:

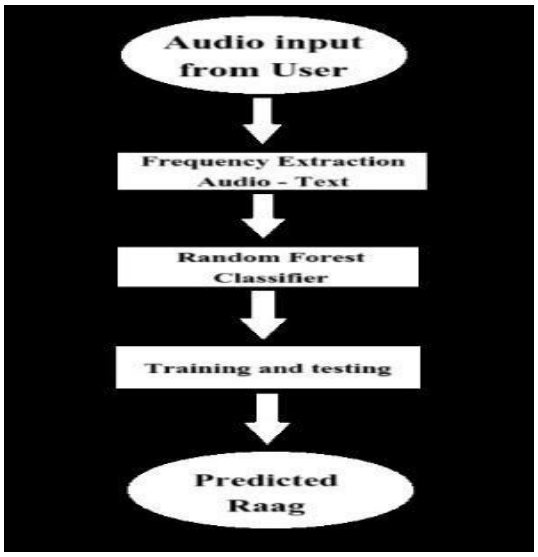


Fig.1. Methodology

1. Data Samples Creation

The research presented in this work is grounded in the creation of a dedicated primary dataset specifically tailored for the task of Hindustani Classical Raag classification. Two Raags — Yaman and Malkauns — were chosen as the focus of the study due to their distinct melodic frameworks and tonal characteristics. For each Raag, 40 individual performances were recorded, giving a total of 80 samples and ensuring a balanced distribution between the two classes.

To control for variations in pitch and eliminate tuning-related inconsistencies, every recording was standardized to the A# tonic. This approach ensured that the model concentrated exclusively on melodic progression, characteristic note patterns, and ornamentation, rather than being influenced by differences in pitch settings between performances.

Data collection was carried out in an acoustically treated studio environment, which significantly reduced the possibility of environmental noise, echoes, or other unwanted acoustic artefacts. A professional-grade microphone setup was used to capture the recordings, providing high clarity and preserving subtle nuances in note articulation, dynamics, and timbre. The Easy Voice Recorder application was selected as the recording tool because of its consistent quality output and reliability in capturing uncompressed audio. All audio files were saved in the .wav format. The decision to use .wav was deliberate — this lossless, uncompressed format retains every detail of the acoustic signal, including microtonal inflections and dynamic contrasts, making it ideal for detailed analysis and feature extraction.

The Raag identification system developed as part of this research allows for direct user input via audio file upload. The system is compatible with both .wav and .mp3 formats. Input files are placed into a dedicated “Audio” folder, from which they are automatically directed into the pre-processing pipeline. In this phase, the audio signal is analyzed and transformed into a structured, text-based representation. The extracted parameters include fundamental frequency trajectories, amplitude variations, pitch contours, note transitions, and other temporal descriptors. Structuring the data in this manner is essential, as it enables the machine learning model to process the audio content as a consistent numerical dataset, facilitating accurate and reproducible classification.

By generating the dataset in-house, the study ensured total control over recording conditions, file quality, and the musical content itself. Each recording was carefully crafted to reflect the theoretical principles of the chosen Raag while incorporating natural variations present in real performances. This approach provided a rich, representative dataset, free from background noise and irrelevant artefacts, while containing clearly identifiable Swar (notes) and Pakad (signature phrases). Such attention to detail not only improved model training and evaluation but also supported better generalization when applying the classifier to new, unseen musical inputs.

2. Pre-processing (Audio to Text Conversion)

The raw .wav recordings were subjected to a multi-step pre-processing pipeline aimed at transforming the unstructured audio signals into a consistent, model-ready dataset. This stage is critical in ensuring that the machine learning model receives uniform, informative inputs, free from unnecessary noise or inconsistencies.

Step 1 – Audio Framing:

Each recording was divided into short, overlapping frames to allow for localized analysis of the signal. This segmentation preserved temporal dynamics, enabling the model to capture fine-grained changes in pitch, amplitude, and harmonic content over time. Frame sizes were selected to balance time resolution with frequency resolution, a key consideration in music signal processing.

Step 2 – Frequency and Temporal Analysis:

For each frame, the frequency content was extracted to generate trajectories that represent how pitch evolves throughout the performance. Additionally, amplitude envelopes were computed to capture intensity variations, while temporal markers (such as onset points) provided cues about note boundaries and transitions.

Step 3 – Feature Structuring and Storage:

Once the acoustic features were computed, they were organized into a structured, time-series format and saved as plain-text files. This choice of storage format made the intermediate data human-readable, easy to debug, and compatible with a variety of analytical tools.

Step 4 – Software Tools and Libraries:

Feature extraction was implemented using Librosa and complementary Python libraries, which provided robust functions for spectral analysis, chroma computation, and statistical summarization. These tools ensured that identical pre-processing steps were applied to both training and testing datasets, eliminating inconsistencies that could lead to biased evaluation.

The classification framework relies on a compact, yet informative set of features derived from the frequency column of each processed audio file. From this data, three statistical descriptors were computed:

Mean Frequency: Represents the average frequency over the entire sample, providing a measure of the tonal center or overall pitch tendency.

Standard Deviation of Frequency: Quantifies the degree of pitch variation, capturing how stable or dynamic the performance is.

Median Frequency: Indicates the middle value of the frequency distribution, serving as a robust central measure less affected by outliers or transient deviations.

These features, while minimal in number, were chosen for their interpretability and their ability to capture the core melodic characteristics of a Raag. The Random Forest Classifier leverages several key mechanisms to enhance classification accuracy and robustness:

1. Bootstrapped Sampling: For each decision tree in the forest, a random subset of the dataset is selected with replacement. This ensures that each tree sees a slightly different training set, promoting diversity in the learned decision boundaries.

2. Random Feature Selection: At each decision node, only a randomly chosen subset of the available features (Mean, Standard Deviation, or Median) is considered for splitting. This prevents all trees from focusing on the same dominant feature and increases the ensemble's overall generalization capability.

3. Independent Tree Training: Each tree is built independently on its respective bootstrapped sample and feature subset. Some trees may rely more heavily on mean frequency, while others might prioritize pitch variation (standard deviation) or central tendency (median).

4. Majority Voting for Classification: Once all trees produce their predictions (either Raag Yaman or Raag Malkauns), the final class label is determined through majority voting — the class receiving the highest number of votes becomes the model's output.

By following this pre-processing approach, the dataset retained the musical essence of each Raag while standardizing the representation for machine learning. This balance between musical fidelity and computational structure was essential for achieving reliable classification results.

3. Training: Random Forest Classier Algorithm

The training and testing of the Random Forest Classifier (RFC) involve multiple steps, including data preparation, model training, evaluation, and performance analysis. In this study, the classifier was trained on extracted frequency parameters from Raag audio samples to accurately identify Raags. The following sections describe the detailed methodology employed in the training and testing phases.

3.1 Data Preparation and Feature Extraction:

The dataset utilized for training the classification model was composed of frequency-based attributes extracted from pre-processed audio recordings of two classical Raags: Yaman and Malkauns. These audio files were first transformed into text format, where each file represented a sequence of frequency values over time. These sequences were then converted into numerical feature vectors suitable for input into machine learning algorithms.

To derive useful and representative features, statistical analysis was performed on the frequency data within each file. Specifically, the mean, median, and standard deviation of the frequency values were calculated. These statistical metrics capture the essential characteristics of the sound profile associated with each Raag and were used as input features for training the model. Additionally, each audio sample was annotated with its corresponding Raag label, which was stored in a structured JSON file (labels.json). This file mapped each filename to its respective Raag, ensuring clarity and organization within the dataset. Prior to model training, the data was divided into two subsets: 40% for training the model and 60% for evaluating its performance (test_size = 0.6). This data split was chosen to test the classifier's ability to generalize well to new, unseen inputs and to avoid the risk of overfitting.

3.2 Model Training using Random Forest Classifier

The Random Forest Classifier (RFC) is an ensemble learning technique that constructs multiple decision trees and aggregates their predictions to improve classification accuracy. It reduces overfitting and enhances performance compared to individual decision trees. Random Forest Classifier is a powerful and widely used machine learning algorithm that belongs to the ensemble learning family. It is an extension of Decision Trees and works by constructing multiple decision trees during training, merging their results to improve accuracy and reduce overfitting.

3.3 Training Procedure:

The classifier was implemented using Sklen: ensemble. Random Forest Classifier with the following hyperparameters:

Number of Trees (n_estimators=100): The model was trained using 100 decision trees, which provided a balance between performance and computational efficiency.

Random State (random_state=42): A fixed random seed was used to ensure reproducibility of results across multiple runs.

Split Criterion (Default: Gini Impurity): The decision trees were trained using the Gini Index, which measures the impurity of splits.

Depth of Trees (max_depth - Not Defined): The depth of each decision tree was left unrestricted to allow the model to learn complex patterns, though it could be adjusted to control overfitting.

Minimum Samples for Split (min_samples_split=2): The model required at least two samples to split a node, ensuring decision boundaries were meaningful

Minimum Samples per Leaf (min_samples_leaf=1): Each leaf node contained at least one sample, allowing the trees to capture detailed variations.

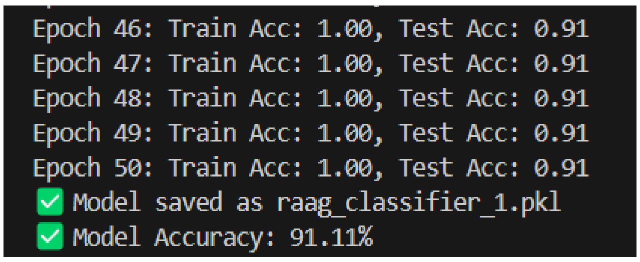
The model was trained over 50 epochs, where it was iteratively refined using training data. In each epoch, the classifier was re-fitted, and its performance was evaluated on both training and test datasets. Since Random Forest does not have an inherent loss function like deep learning models, a dummy loss function was defined as the error rate (1 - accuracy). This helped the model to track how well the model was learning over epochs.

3.4 Model Testing and Evaluation

Once trained, the model was tested on the reserved test dataset, and its performance was assessed using the following evaluation metrics:

Accuracy Score: The classifier achieved 91% accuracy, which indicates a strong learning capability. Accuracy was calculated using accuracy_score(y_test, y_pred), where y_test represents the actual labels and y_pred represents the predicted labels. The accuracy was computed using the accuracy_score function from the sklearn. metrics module. The formula for accuracy is:

$$\text{Accuracy} = \mathit{Total\ Number\ of\ Correct\ Predictions/Total\ number\ of\ Predictions} \quad (1)$$



```
Epoch 46: Train Acc: 1.00, Test Acc: 0.91
Epoch 47: Train Acc: 1.00, Test Acc: 0.91
Epoch 48: Train Acc: 1.00, Test Acc: 0.91
Epoch 49: Train Acc: 1.00, Test Acc: 0.91
Epoch 50: Train Acc: 1.00, Test Acc: 0.91
✅ Model saved as raag_classifier_1.pkl
✅ Model Accuracy: 91.11%
```

Fig.2. Model Training Results

Confusion Matrix: To assess the performance of the trained Random Forest model, a confusion matrix was created to offer a detailed breakdown of how accurately the model classifies each Raag. This matrix serves as a valuable tool for comparing the model's predicted labels with the actual labels, allowing us to evaluate its performance and identify the types of errors it makes.

The confusion matrix includes the following elements True Positives (TP), represents the instances where the model correctly identified a Raag. False Positives (FP), where the model mistakenly predicted a Raag that was not present. False Negatives (FN), where the model failed to identify a Raag that was present. True Negatives (TN), where the model correctly identified the absence of a Raag. By evaluating these components, we can calculate the model's overall accuracy and pinpoint specific areas where it is prone to making errors, such as Raags that are more likely to be misclassified.

Heatmap Visualization: To enhance the interpretability of the confusion matrix, a heatmap was generated using a popular Python visualization library, seaborn. It gives a visual representation of the confusion matrix, where different colors are used to indicate each class of prediction. The color intensity or shade in each cell corresponds to the number of samples in specific category. This color-coded representation makes it easy to identify trends and patterns. Darker shades signify higher values, showing areas where the model was more accurate or made more frequent misclassifications. Lighter shades indicate lower values, highlighting instances where errors are less common. By visualizing the confusion matrix as a heatmap, it becomes easier to detect which Raags the model tends to misclassify and identify any patterns or biases that could be addressed.

Interpretation: Examining both the confusion matrix and the heatmap provides valuable insights into the model's performance across different Raags. This analysis is crucial for identifying areas where the model may be underperforming or exhibiting biases, allowing for targeted improvements in the classification process. It also guides potential adjustments to the model, such as refining feature extraction methods, tuning hyperparameters, or exploring different algorithms to enhance accuracy in future iterations.

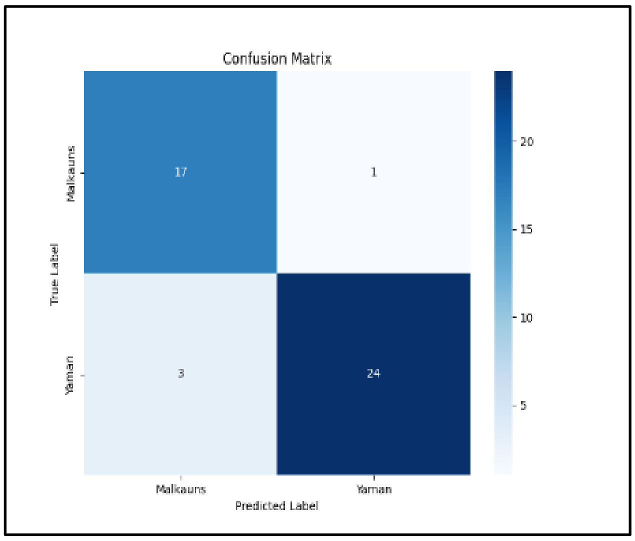


Fig.3 Confusion Matrix

To better understand the learning behavior of the model, the following plots were generated. The plots are generated on the basis of epochs for the training. One epoch indicates that the complete training dataset is passed through the model learning algorithm. It plays vital role to understand how a machine extract information and learns from the data provided. It helps to adjust the weights and biases for the training enhancing the accuracy and gradual reduction in errors. Here these graphs give an idea of how the accuracy and loss parameters changes per epoch.

Training vs. Testing Accuracy Graph: It is important to differentiate the training and testing data to avoid overfitting. The datasets are loaded in three steps, Supply (allocate data to the model), Declare (Conversion of data to model suitable format), Run (put the model for testing). Plotted accuracy over 50 epochs to visualize performance improvements. Showed how well the model generalized to unseen data.

Training vs. Testing Loss Graph: Since Random Forest does not compute loss directly, error rate (1 - accuracy) was used as a substitute for loss. The loss graph helped identify potential overfitting issues. ‘If the training error is very low but the testing error is significantly higher, it indicates that the model is fitting the training data too closely and not generalizing well to unseen data; a classic sign of overfitting’ [16]. On the other hand, if both training and testing errors are high, it may indicate underfitting, suggesting the model is too simple or insufficiently trained.

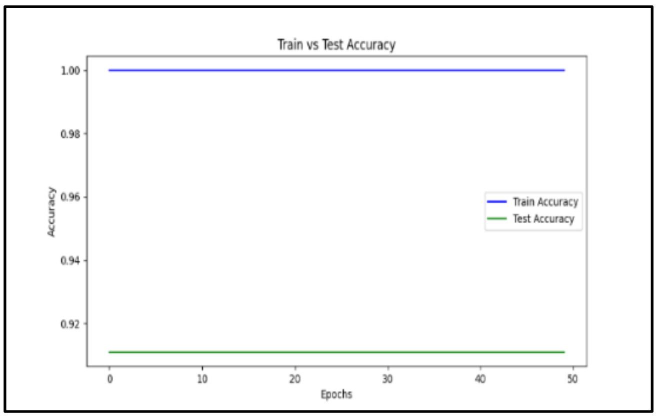


Fig. 4 (a) Accuracy Graph for Train vs. Test

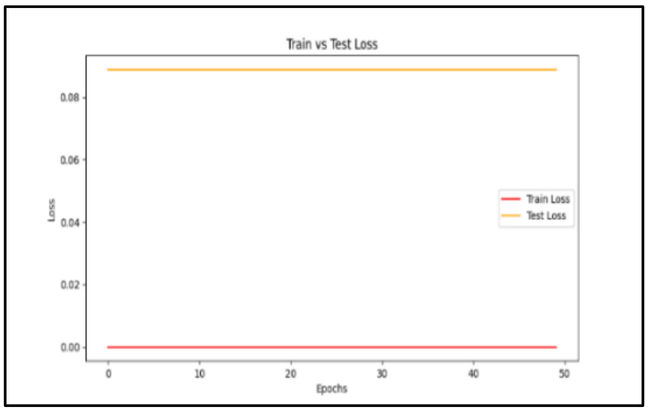


Fig.4 (b) Loss Graph for Train vs. Test

4. Testing Process for Raag Classification Model:

The trained Random Forest Classifier (RFC) was tested using new audio samples, which were pre-processed into text files containing extracted frequency parameters. This section details the testing pipeline, including feature extraction, model loading, and prediction steps.

In the first step in the testing pipeline involved processing the new audio samples. Each audio file was pre- processed to extract relevant frequency parameters, which were stored in text files. The features extracted from the audio signals were consistent with those used during the training phase to maintain compatibility. These features typically include statistical measures such as the mean frequency, standard

deviation, and median frequency, which were computed for each audio sample.
Once the features were extracted, the trained Random Forest Classifier (RFC) model was loaded into memory. The model, which had been previously trained on a labeled dataset, was serialized and saved for later use. During testing, the saved model was loaded into the pipeline to make predictions based on the new test data. This
dynamics), and ABM (agent-based modelling). These approaches are essential to study the mechanical response of tissues, single cells, and extracellular substances under different stimuli. Prendergast [1] investigates the effects of mechanical loading on tissue formation by employing finite element Modelling to simulate stress-induced growth. Mak et al. [2] reveal the force of multiscale integration by coupling molecular dynamics with tissue-level mechanics to better understand cellular responses considered as a whole.

Tepole [3] presents a methodology based on systems biomechanics for Modelling wound healing, emphasizing the complexities of mechanical-biochemical feedback in regenerating tissues.Considering bone mechanobiology by in silico Modelling, Giorgi et al. [4] provide some insights into the prediction of fracture healing and osteo-interventions.Rajagopal et al. [5] delve into the mechanical deformation of individual cells, utilizing computational tools to assess cytoskeletal behavior under stress. Their models help in understanding mechanosensitivity and cell signalling pathways.

Newer advances build on these prior technics and frameworks. Brown et al. [6] use computational models to explore cardiac development and investigate the role of mechanical cues in shaping morphogenesis and in congenital heart disease. Boaretti et al. [7] advance the perspective on multicellular in silico models in studying coordinated cellular mechanisms in bone reModelling. Dolan et al. [8] present a summary of a suite of techniques—force microscopy, image-based Modelling—used to explore mechanotransduction at various spatial scales.

Together, these studies demonstrate how computational Modelling has transformed mechanobiology. Through simulating mechanical environments with great accuracy, they facilitate linking experimental data to theory and promoting developments in personalized medicine, regenerative therapies, and target-specific intervention design. With mechanobiology increasingly interfacing with artificial intelligence and high-throughput data analytics, these tools will find increasingly more use in biomedical exploration.

ensures that the model remains consistent with its training parameters.
With the extracted features and the loaded model, the final step was to pass the feature vectors from the test data into the RFC. The model then processed the feature vectors and generated predictions regarding the class or category of the audio samples. The RFC leverages the decision trees that were created during the training phase to assess the test data and output the predicted class label.
Each new audio sample was processed through this pipeline, allowing for efficient and automated predictions. This approach ensures that the classifier could be reliably applied to unseen data, providing consistent results in real- world scenarios.

4.1 Loading the Trained Model

Before testing new audio samples, the previously trained Random Forest model was loaded from a saved file using the joblib.load() function. The model had been serialized and saved as raag_classifier_1.pkl during the training phase. By using this method, the model could be quickly and efficiently reloaded without the need for retraining, ensuring both time and computational efficiency. The joblib.load() function was used to deserialize the model from the file, restoring its internal structure, including the trained decision trees and their corresponding parameters. This process re-establishes the Random Forest Classifier (RFC) in its previous state, making it ready for use in making predictions on new data.
By using the saved model, we ensure that the classification system is both reusable and consistent. This method eliminates the need for redundant training, allowing predictions to be made directly from the test data. It also ensures that the classifier remains aligned with the original training setup, preserving its performance without any modifications or adjustments to the model. This approach streamlines the prediction process, making it more efficient and practical for real-world deployment.

4.2 Processing New Test Data

The test dataset comprises audio recordings that have been processed using the same feature extraction and transformation pipeline as the training data, ensuring consistency in input format and structure. These audio files are first converted into text format and saved within a specific directory labeled Test_Text.
Each of these text files encapsulates frequency-related attributes derived from the original audio, such as pitch variations and note dynamics, which serve as inputs to the classifier. Before classification, every test file undergoes a series of validation steps to ensure the integrity and reliability of the data being passed into the model:
File Reading Check: Each file is loaded using the pandas.read_csv() function. This step checks whether the file is readable and properly formatted. In cases where a file fails to load due to encoding issues, corruption, or improper formatting it is excluded from further processing to maintain the robustness of the pipeline.

Column Presence Verification: A crucial check is performed to confirm the existence of the "Frequency (Hz)" column within each file. This column contains the core data required for generating meaningful statistical features (e.g., mean, standard deviation). If this column is missing, the file is automatically skipped to avoid feeding incomplete or incorrect data to the machine learning model.
By enforcing these validation steps, the pipeline ensures that only high-quality, well-structured test data is used in the evaluation phase, thereby enhancing the reliability and accuracy of the raag classification results.

4.3 Feature Extraction for Prediction

To maintain consistency with the training phase, the same set of statistical features was extracted from the frequency data to ensure that the model receives comparable input during both training and testing. Specifically, the following statistical measures were used:

Mean Frequency (np.mean(frequencies)): This measure represents the average of the frequency values, providing a central point around which the frequencies are distributed. It is a fundamental metric for understanding the general tendency or "center" of the frequency distribution.

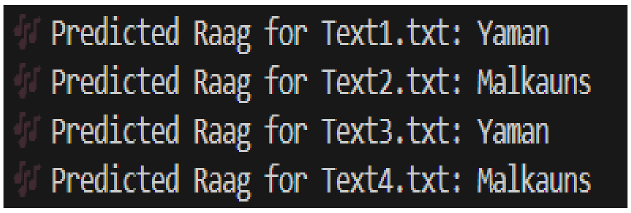
Standard Deviation (np.std(frequencies)): This value quantifies the extent of variation or dispersion within the frequency data. A higher standard deviation indicates a wider spread of frequencies, while a lower value suggests that the frequencies are more concentrated around the mean.
Median Frequency (np.median(frequencies)): The median serves as a robust alternative to the mean, offering a central value that is less sensitive to outliers. It divides the frequency data into two equal halves, making it especially useful when there are extreme values that could skew the mean.
These extracted statistical features were compiled into a feature vector, which was then reshaped into a one-dimensional array. This reshaping ensures that the data format aligns with the input requirements of the trained model, enabling smooth integration of the test data with the model's expected structure. By maintaining this consistent approach, the model can evaluate the test data in the same manner as the training data, ensuring valid and reliable predictions.

4.4 Raag Prediction using Random Forest Classifier:

After extracting the relevant features and forming the feature vector from each test sample, this vector is fed into the previously trained Random Forest classifier using the clf.predict(feature_vector) method. The classifier, having already learned the distinguishing patterns and statistical traits of each raag during the training phase, analyzes the input and determines the most likely raag associated with the feature set.
The prediction result is a raag label corresponding to the input sample, which reflects the classifier's interpretation based on the learned characteristics of the training data. This output is then recorded or displayed for each test sample, effectively completing the classification process.

4.5 Results and Output Interpretation

For every test sample processed, the model prints a predicted Raag label along with the corresponding filename. In addition to displaying the predicted Raag alongside the filename, this output format makes it straightforward to verify the model's performance on individual test samples. This provides an



```
Predicted Raag for Text1.txt: Yaman
Predicted Raag for Text2.txt: Malkauns
Predicted Raag for Text3.txt: Yaman
Predicted Raag for Text4.txt: Malkauns
```

Fig. 5 Testing Results for various Samples

easy-to-interpret classification output for new audio samples. The structured output follows this format given in fig 5:

Each prediction reflects the classifier's interpretation of the input frequency features derived from the corresponding audio segment. This structured feedback is especially helpful during evaluation, as it allows for easy comparison with ground truth labels and facilitates error analysis.
Consistent and accurate predictions like those shown in the figure help demonstrate the effectiveness of the model in distinguishing between Raag Yaman and Raag Malkauns. Furthermore, this type of output can be seamlessly extended to real-time applications, such as music identification systems or digital Raag learning tools, where interpretability is essential.

5. CONCLUSION:

This study demonstrates that frequency-based statistical features, when paired with a Random Forest Classifier, can effectively distinguish between Raag Yaman and Raag Malkauns with an accuracy of 91%. Despite the complexity of Hindustani Classical Raags, the results indicate that a minimal feature set, if carefully selected, can yield robust classification performance without requiring deep neural networks or large datasets.
The proposed approach benefits from the interpretability and low computational cost of ensemble learning, making it suitable for integration into music education tools, digital archiving systems, and real-time performance analysis platforms. By standardizing all recordings to a single pitch (A#), the model avoids confounding tonal variations, focusing purely on melodic and structural features.
Future work will address current limitations by:
Expanding the dataset to include multiple Raags across diverse pitch settings and instruments.
Incorporating pitch-independent representations to enhance generalization.
Exploring temporal and spectral features for finer-grained analysis.
Comparing RFC performance with deep learning architectures such as CNNs and RNNs.
Through these enhancements, the system can evolve into a comprehensive AI-assisted platform for Hindustani music analysis, bridging the gap between computational methods and traditional musicology.

6. FUTURE SCOPE:

Despite the promising results obtained in this study, several limitations must be acknowledged to provide a comprehensive view of the work. Firstly, the scope of the study was deliberately confined to a specific pitch range, which, while beneficial for controlled experimentation, restricts the model's ability to generalize across diverse vocal or instrumental registers. This may limit its applicability in real-world scenarios where pitch can vary significantly between artists, instruments, and performance styles.
Secondly, the number of Raags included in the classification task was limited, focusing only on Raag Yaman and Raag Malkauns. While this helped in building a focused and well-and richness of Hindustani classical music, which consists of

hundreds of Raags, each with subtle and unique characteristics. Expanding the dataset to include a broader range of Raags would significantly enhance the robustness and generalizability of the model.

Future research can build upon this foundational work by incorporating a larger and more diverse set of Raags, introducing variations in tempo, rhythm, and performance style. Additionally, improvements in audio pre- processing techniques, such as advanced noise reduction algorithms, dynamic range compression, and harmonic separation, can help refine feature extraction from raw audio inputs.

Another vital area for enhancement is the integration of Swar (note) identification, which plays a key role in characterizing Raags. By identifying the exact notes and their patterns, the model could achieve a deeper understanding of melodic structure. Moreover, developing pitch-independent classification techniques would allow the model to accurately identify Raags regardless of the tonic scale used by the performer, making it more versatile across different vocal and instrumental renditions.

Overall, this study lays the groundwork for more sophisticated systems capable of capturing the intricate nuances of Hindustani classical music, paving the way for applications in music education, archival research, and real-time music analysis.

REFERANCES:

1. Author: Dr. Vidyadhar Oke, “22 Shrutis and melodium”, Sanskar Prakashan, 2011th edition.
2. Author: Acharya S. N. Ratanjankar, “Aesthetic Aspects of India’s Musical heritage”, Sanskar Prakashan, first edition Feb 1992.
3. Changsheng Xu, N. C. Maddage, Xi Shao, Fang Cao, Qi Tian. “Music genre classification using Support Vector Machines”, IEEE International Conference in Acoustic, Speech and Signal Processing (ICASSP’03), 2003.
4. Ndiatenda Ndou, Ritesh Ajoodha, Ashwini Jadhav. “Music Genre Classification: A Review of Deep-Learning anf Traditional Machine-Learning Approaches’, IEEE International IOT, Electronics and Ma=echatronics Conference (IEMTRONICS), 2021.
5. N. Scaringella, G. Zola, D. Mlynek. “Automatic Genre Classification of music content: a survey”, IEEE Signal Processing Magzine (Vol:23, Iss:2), 2006.
6. Aakash Parmar, Rakesh Katariya, Vatsal Patel. “A Review on random Forest: An Ensemble Classifier”, International Conference on Intelligence Data Communication Technologies and Internet of Things (ICICI), 2018.
7. M. Pal. “Random Forest classifier for remote sensing classification”, International Journal of Remote Sensing (Vol:26), 2005.
8. Archana Chaudhary, Savita Kolhe, Raj Kamal. “An Improved random forest classifier for multi-class classification”, Information Processing in Agriculture, Elsevier, (Vol:3, Iss:4) 2016.
9. Md. Rifat Ansari, Sadia Alam Timpa, Jannat Ara Ferdouse Raya, Mohammad N. Murshed. “Comparison between Support Vector Machine and Random Forest for Audio Classification”, International Conference on Electronics, Communications and Information Technology (ICECIT), 2021.
10. Baoxun Xu, Xiufeng Guo, Yumming Ye, Jiefeng Cheng. “An Improved Random Forest Classifier for Text Categorization”, Journal of Computers (Vol:7 Iss:12) Academy Publisher, 2012.
11. Zehra Cataltepe, Yusuf Yaslan and Adullah Sonmez. “Music Genre Classification using MIDI and Audio Features”, EURASIP Journal on Advances in Signal Processing, Volume 2007, Hindawi Publishing Corporation, Article ID: 36409
12. A. Elbir and N. Aydin. “Music Genre Classification and Music Recommendation by using Deep Learning”, Institution of Engineering and Technology Wiley Online Library, 2025.
13. Michael Haggblade, Yang Hong, Kenny Kao. “Music Genre Classification”. https://cs229.stanford.edu/proj2011/HaggbladeHongKao-MusicGenreClassification.pdf
14. Juhan Nam, Keunwoo Chol, Jongpil Lee, Szu-Yu Chou, Yi-Hsuan Yang. “Deep learning for audio-based music classification and Tagging: Teaching computers to distinguish Rock from Bach”, IEEE Signal Processing magazine (Volume 36, Issue 1), 2019
15. Jyoti Lele, Naman Palliwal, Sahil Rajurkar, Vibor Tomar and Anuradha C Phadke, “Comparison of image

based and audio-based techniques for Bird-species Identification", Ed. Intelligent Systems and Applications in Computer Vision, CRC Press- Taylor Francis Publication, 2023, PP. 9-1 to 9-10 pages, Book ISBN9781003453406

16. Zhihao Jia, Sina Lin, Mingyu Gao, Matei Zaharia, Alex Aiken. "Improving the Accuracy, Scalability and Performance of Graph Neural Networks with Roc", Proceedings of Machine Learning and Systems (MLSys 2020).

11. Zehra Cataltepe, Yusuf Yaslan and Adullah Sonmez. "Music Genre Classification using MIDI and Audio Features", EURASIP Journal on Advances in Signal Processing, Volume 2007, Hindawi Publishing Corporation, Article ID: 36409