# Classifying Carnatic Ragams Based On Audio Features

**Ananya Srinivasan**
Belmont, MA, United States

## ABSTRACT

Carnatic music is a form of classical singing originating from South India. It consists of many nuances and patterns that are organized into various ragams, which are melodic frameworks for a song that dictate the types of notes and modulations present. Each ragam contains complicated variations of notes and features, however, so it is often difficult for singers and listeners to identify ragams. To address this challenge, I built a model using a Random Forest Classifier that analyzes features such as Chroma Short-Time Fourier Transform, Spectral Centroid, Spectral Bandwidth, various MFCC coefficients, and the ragam label in order to identify the ragam of the audio clip. The model was trained on a dataset of 1112 audio clips that includes 32 ragams– 7 melakarta "parent" ragams and 25 janya "child" ragams (Sanjana Satish68l). Each ragam has certain patterns within its features that make it easy to distinguish from other ragams, and therefore easy for the model to categorize. By analyzing these distinct features, the model was able to accurately classify the ragam of almost all the test clips. The accuracy was considerably high: around 99.1%, which outperformed both the K Nearest Neighbors and Decision Tree models.

## INTRODUCTION

Many people can empathize with and enjoy music. They find it simple to identify different genres, rhythms, and melodies of songs. However, in certain types of music, it may be harder to distinguish between the various nuances, such as in an Indian classical form of singing called Carnatic music. It's a singing form that originated in South India and dates as far back as the 12th century. Carnatic compositions contain many nuances, such as voice modulations between notes called gamakas, and are organized based on their ragam and talam. A talam is a rhythmic beat that measures time in terms of the song. It occurs in cycles of beats and is kept track of using the person's hands. There are various unique talams, each with a specific number of beats per cycle. A ragam is a melodic framework for the song; it includes notes called swaras, specific patterns of these notes, and voice modulations that must be present in the song. Carnatic music is based on seven notes, called swaras: Sa, Ri, Ga, Ma, Pa, Da, and Ni. These notes have specific frequencies and pitches and can be sung in a variety of ways. For example, there are multiple versions of Ri (Ri 1, Ri 2, and Ri 3) as well as for Ga, Ma, Da, and Ni, all of which vary in pitch. Each ragam consists of an ascending (arohanam) and descending (avarohanam) set of swaras that dictate the patterns of notes present in the ragam. There are 72 melakarta or "parent" ragams; these ragams must include all 7 swaras in the correct ascending and descending order, though with variations in the gamakas and pitch of each swaram. There are hundreds more janya ragams, or "child" ragams, which include at least 5 swarams but can be in varying orders and are derived from these melakarta ragams. Each of these ragams has intricate nuances that can sometimes be difficult for even professional singers to tell apart, especially if the singer is only singing the lyrics and not the notes themselves. Identifying these ragams requires extensive training in or exposure to Carnatic music. Many people enjoy listening to Carnatic music but aren't able to distinguish the ragam by just hearing the song, so they're unable to enjoy the subtle patterns present in each ragam. Furthermore, it would help if Carnatic students were able to identify ragams and their key characteristics without decades of experience. My research aims to address the matter of identifying and classifying the ragam of a Carnatic song using its audio alone, without the need for professional assistance. Classifying ragams by just hearing the song can help people who may not be experts understand the nuances of each ragam, allowing them to appreciate the music more. It can help them connect with the music and its meaning, and it's beneficial for people who are learning Carnatic music and people who love to listen to it.

In 2018, V. Kaimal and S. Barde of the Dept. MATS School of Information Technology in Raipur, India proposed a methodology for obtaining the audio's frequencies and matching it to the appropriate ragam to classify (Kaimal and Barde). The audio would be analyzed to find the particular frequencies it consisted of, and these frequencies would be matched to the

frequencies of various swaras that were stored in the database. The model was also trained with a tool called Praat, which analyzed a database of ragams and identified key features for each. To identify the ragam of the audio, the swara patterns identified using the frequency would be matched to the information about the ragams the model was trained on, allowing it to classify the ragam from the audio. It also predicts the Hindustani ragam along with the Carnatic one, and in the end uses various classifiers to calculate the accuracy of the prediction.

In 2009, R. Sridhar and T.V. Geetha of Anna University in Chennai, India, proposed a methodology that breaks the audio into segments and analyzes each segment to gain the necessary information to classify it (Sridhar and Geetha). The audio is first divided into smaller segments, and the singer's voice is separated from the instruments. These segmented clips are then checked for the frequency components that correspond to each swara. In this case, the segmented audio was also used to identify the song's talam. They identified the onset (start) of the audio (where the amplitude of the frequency first rises from zero) and the offset (where the audio ends) and divided the audio. A model trained with a dataset of 175 different talams then matches the number of cyclic beats in the audio segment to its corresponding talam. After this, the model identifies the frequency components in each segment and matches these patterns with the frequency patterns of corresponding swaras. Specifically, the frequencies are compared to frequencies of swaras sung by various singers, data which the model stores and is trained on. This identifies the swaras present in the audio. The swaras are then compared to the ragam patterns of swaras stored in another database to match the swara patterns with the ragam to identify them.

A paper by S. Natesan and H. Beigi in May 2024 proposes the methodology of using a Neural Network model to classify the notes based on their frequencies and identify the ragam (Natesan and Beigi). It utilizes Discrete Fourier transformation and Triangular Filtering systems to identify features and frequencies of the sounds present and create groups of possible notes. The model is trained using a dataset of 676 different recordings to identify the swara patterns and the ragam using a combination of Convolutional Neural Networks and Recurring Neural Networks. In order to account for the differences in the singer's pitch, the model compares the features of each note relative to other notes rather than absolute differences.

A paper by B.T. Rao, S. Chinnam, P.L. Kanth, and M. Gargi of The Science and Information Organization in December 2012 proposes the methodology of using a K Nearest Neighbors model to identify the melakarta ragam of the audio (Rao et al.). It compares the features of the audio to various closest matches, or 'neighbors,' and uses these to determine the ragam. The paper explains different ways to measure the distance between neighbors, like Cosine Distance and Earth Movers Distance, and discusses how the algorithm works for training and testing. The results show that Earth Movers Distance gives better accuracy than Cosine Distance for the nearest neighbors model.

All these past studies show that the key to classifying ragams is identifying the unique audio features such as frequency for each ragam. This paper introduces the proposed machine-learning model that uses a Random Forest Classifier trained on factors such as spectral bandwidth and mfcc coefficients to classify the Carnatic ragam of an audio clip accurately.

## DATA

The dataset used is from a public dataset on Kaggle called "Ragas with features in Indian classical music" (Sanjana Satish68l). The dataset consists of 1112 30-second audio files and contains information about several audio features for each file. The rows of the dataset are each file, and the columns represent different features such as Chroma Short-Time Fourier Transform (Chroma STFT), Spectral Centroid (spec_cent), Spectral Bandwidth (spec_bw), various MFCC coefficients, and the ragam label (see Figure 1). Chroma STFT is a feature used to describe the tonal quality and content of the audio, meaning the kinds of tones it contains. Specifically, it's used to represent information about the classification of the pitch and signal structure of the audio. Spectral Centroid (spec_cent) is a feature that categorizes the audio's spectrum. It indicates where the center of mass of the spectrum is; in simple terms, where most of the "energy" of the audio is located, which is calculated with an amplitude-weighted mean. The spectral centroid is used to measure the perceived "brightness" of a sound, so a higher spectral centroid value corresponds with a "brighter" sound. The feature of Spectral Bandwidth (spec_bw) measures the range of frequencies in the sound by finding the difference between the highest and lowest frequencies present.

Spectral bandwidth generally corresponds with the perceived timbre, or certain sound qualities unique to different instruments or melodies that can help identify them. Lastly, MFCCs– Mel-Frequency Cepstral Coefficients– are a small set of features, usually about 10-20, that describe the overall shape of the spectral envelope. A spectral envelope is the shape of the power spectrum of the audio, and similar to the spectral bandwidth is an important cue for identifying certain sounds, such as voices or different instruments. These audio features are extremely significant because the model is trained by analyzing the audio features, rather than using each audio file itself. The dataset includes audio files for 32 popular ragams: "Amritavarshini", "Anandabhairavi", "Bhairavi", "Bilahari", "Charukesi", "Darbari kannada", "Gaula", "Hamsadhwani", "Hamsanandam", "Harikambhoji", "Hindolam", "Jaganmohini", "Janaranjani", "Kalyani", "Kamas", "Kamboji", "Kanada", "Kapi", "Karaharapriya", "Kedaram", "Madhyamavati", "Mayamalavagaula", "Mohanam", "Nata", "Ranjani", "Sahana", "Saveri", "Shankarabharam", "Sindhubhairavi", "Sri Ranjini", "Todi" and "Varali". It includes both melakarta (“parent”) ragams and janya (“child”) ragams. There are 7 melakarta ragams in this dataset, and 25 janya ragams. However, there are a lot more audio files of certain ragams than others in this dataset. For example, there are around 480 files for the ragam Kapi, but below 50 files for the ragam Todi (see Figure 2). The distribution of the ragams isn’t even. Although the model is still able to classify the ragams based on their features, this could potentially be the source of bias and possible errors, as discussed later.

| | raga | Unnamed: 0 | rmse | chroma_stft | spec_cent | spec_bw | mfcc0 | mfcc1 | mfcc2 | mfcc3 | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Amritavarshini | 110.0 | 0.018164 | 0.292356 | 1558.750051 | 1016.731450 | -350.829612 | 134.589973 | -45.223985 | 7.677200 | ... |
| 1 | Anandabhairavi | 150.0 | 0.018164 | 0.292356 | 1529.762912 | 1016.731450 | -368.231472 | 149.686020 | -83.656807 | 3.760192 | ... |
| 2 | Bhairavi | 349.5 | 0.027011 | 0.265930 | 1326.192217 | 1119.476952 | -398.466602 | 164.589604 | -63.290213 | -6.427252 | ... |
| 3 | Bilahari | 123.5 | 0.018164 | 0.292356 | 1741.067286 | 1016.731450 | -300.369497 | 129.661850 | -53.972727 | 15.576789 | ... |
| 4 | Charukesi | 97.0 | 0.018164 | 0.292356 | 1571.460616 | 1016.731450 | -331.383352 | 138.423632 | -58.595126 | 3.475179 | ... |

5 rows × 25 columns

**Figure 1:** First 5 rows of data from the “Ragas with features in Indian classical music” dataset. Includes chroma_stft, spec_cent, spec_bw, and 4 out of the 19 MFCC coefficient data

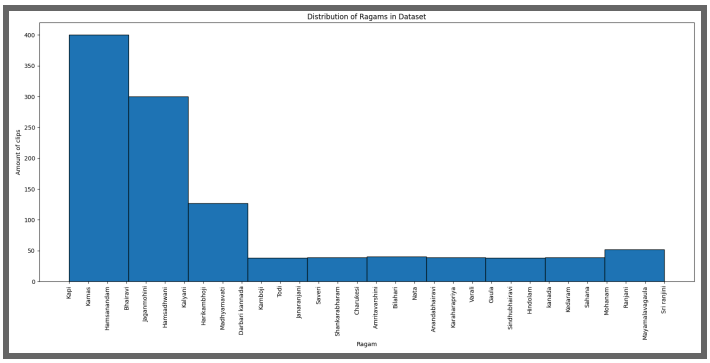


**Figure 2:** Distribution of ragams in the “Ragas with features in Indian classical music” dataset

Certain audio features distinguish these ragams. The model is trained to analyze these patterns and use it to classify the ragam based on the audio features. Most features– chroma STFT, spec_cent, spec_bw, and MFCCs have largely unique or identifiable values for each of the ragams. For example, the values for MFCC10 seem to be unique for every ragam. Evey ragam has a relatively distinguishing value (see Figure 3), so the model would likely find it easy to categorize the ragam based on its unique MFCC10 features. In regards to spectral bandwidth (spec_bw), most of the ragams have a similar value. However, there are a few that stand out: Kapi, Kamas, Kalyani, Jaganmohini, Harikambhoji, Hamsanandam, Hamsadhwani, and Bhairavi (see Figure 4). These are the only ragams with spectral bandwidth values that are above or below the others. Patterns like these could be used to distinguish the ragam by comparing the spectral bandwidths of the audio features. Lastly, the MFCC8 graph is another example of patterns of audio features. All the ragams have a negative value for MFCC8; however, Kamboji alone has a positive MFCC8 value (see Figure 5). In this case, it can be used to identify the ragam Kamboji. Similarly, patterns like these can be used to identify specific ragams with features that stand out from the others.

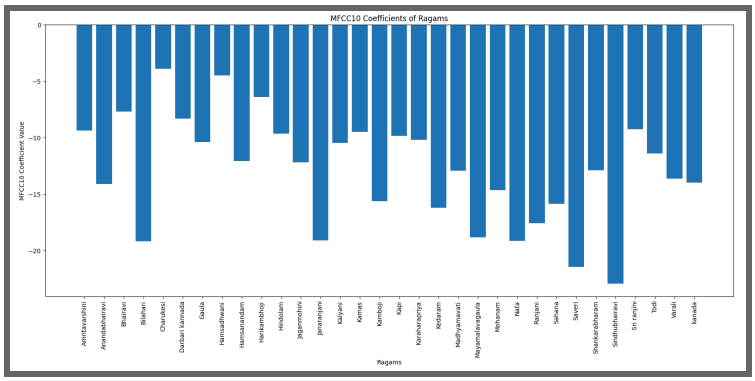


**Figure 3:** Visualization of the MFCC10 values of each ragam in the dataset

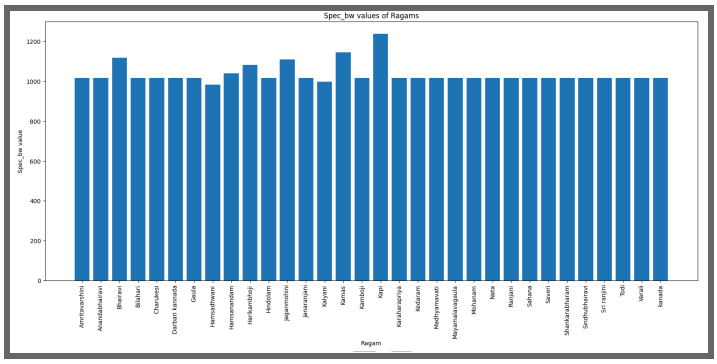


**Figure 4:** Visualization of the Spectral Bandwidth (spec_bw) of each ragam in the dataset

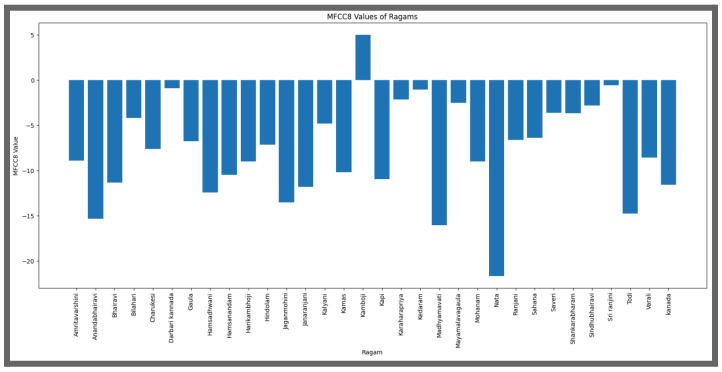


**Figure 5:** Visualization of the MFCC8 values of each ragam in the dataset

## MODEL

Before developing the model, the first step was splitting the data. I split the training and testing data using a test size of 0.2, so the training-testing split was 80-20: 80% of the data was assigned as training data, and 20% was assigned as testing data. In order to make the model able to predict the ragam, I turned the y values– currently the ragam itself– into numerical values that the computer could deal with. I used the label encoder function from sklearn to convert the ragam names into numbers: the number 1 corresponded to a certain ragam, and 2 to a different one. The model I chose to classify the ragams is a Random Forest Classifier. A random forest is a method that consists of multiple decision trees working together to classify data or make predictions (see Figure 6). Each decision tree starts by asking a series of questions about the different features present in the dataset. These questions help the tree split the data into branches and, finally, generate a final prediction based on the answers it receives. What makes a random forest effective is its use of multiple sets of decision trees. Instead of relying on just one tree, the random forest combines the predictions from all its trees. This collective approach helps improve the overall accuracy of the model and reduces the chance of errors. Each decision tree in the forest is built using a random part of the data. This randomness makes sure that the trees are diverse, analyzing different patterns or aspects of the data. This makes the random forest more reliable than any single decision tree alone.

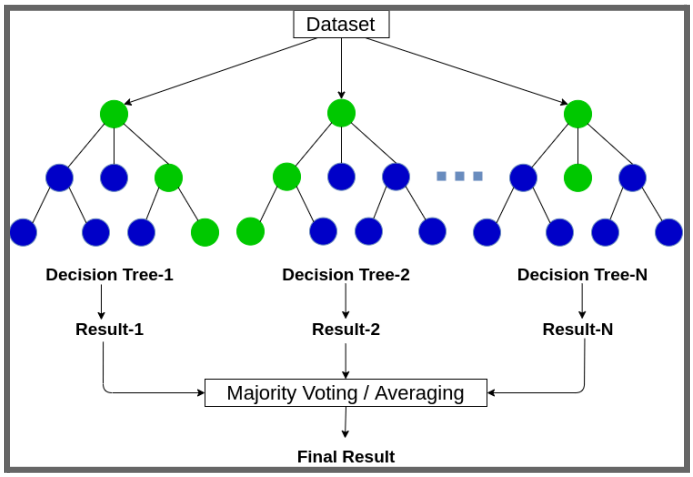


**Figure 6:** Visualization of how a Random Forest Classifier works

To build the model, I hyperparameter-tuned the Random Forest Classifier to determine its ideal max depth. The max depth determines how many splits each decision tree is allowed to make. I tested different max depths from 1 to 20 and graphed the accuracy over this interval. The graph indicated that the ideal max depth for the random forest classifier is 17 (see Figure 7).

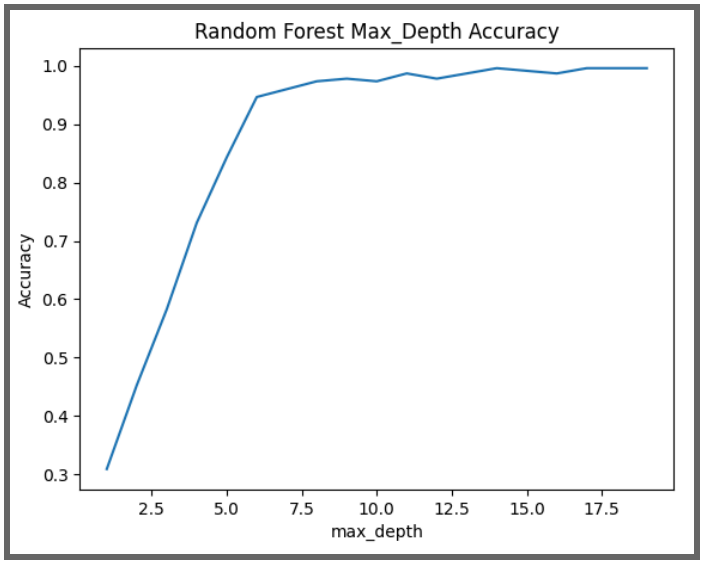


**Figure 7:** Accuracies of the Random Forest Classifier model plotted against various max_depth values

I then initialized the Random Forest Classifier model with a max depth of 17. The model was fitted to the X and y training data and used to predict the ragams of the X testing data. To determine the accuracy of the model, the predictions were compared to the actual ragam label (y test) and graphed for visualization (see Figure 8).

Before I chose the Random Forest Classifier, I also experimented with a couple of baseline models including K Nearest Neighbors classification and the Decision Tree Classifier. The K Nearest Neighbors Classifier was hyperparameter-tuned, and according to the accuracy graph, its ideal number of neighbors for this dataset was 1. The Decision Tree Classifier was also hyperparameter-tuned and had an ideal depth of 15. Neither of these yielded a better performance than the Random Forest Classifier, however (the results will be discussed later).

## RESULTS

The model with the highest accuracy was the Random Forest Classifier model. Through hyperparameter tuning, I determined that the ideal max depth of the random forest was 17. Using this max depth, I tested the model on the X-testing data and plotted the accuracy. The accuracy of the model was 99.1%, which was higher than both the Decision Tree Classifier (95.5%) and the K Nearest Neighbors Classifier (92.4%). This indicates that the predictions of the Random Forest model were near perfect (see Figure 8). The Decision Tree's performance was slightly lower than the Random Forest, with an accuracy of 95.5%. The K Nearest Neighbors model, which performed worse than the Decision Tree and the Random Forest, had an accuracy of 92.4%.

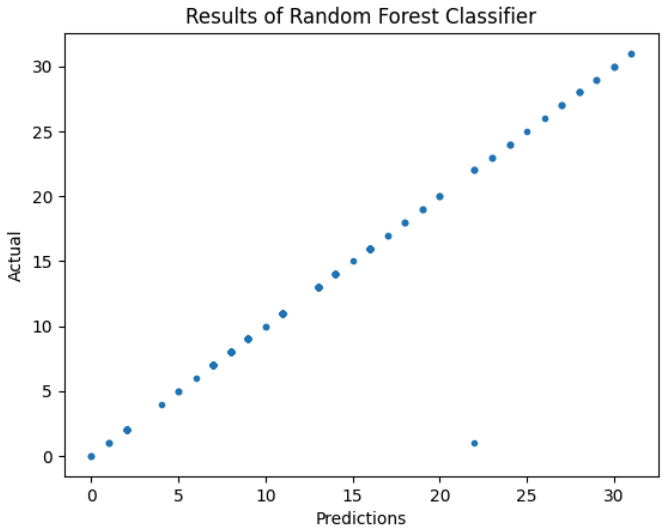


**Figure 8:** Graph comparing the predictions made by the Random Forest Classifier with the actual values of the testing data

## DISCUSSION

The Random Forest model was incredibly accurate– 99.1 %, which is near perfect. This means that it accurately guessed the ragam of the test data 99.1% of the time based on the features. As discussed in the data section, many unique features help distinguish each ragam, so it's likely easy for the model to categorize based on the features. Each ragam seems to have its own set of distinct patterns; the individual ragams aren't very similar. This could also potentially be why the K Nearest Neighbor model didn't work as well as the other two. K Nearest Neighbor model looks for a certain number of "neighbors" or close matches of the audio features to determine the ragam. During hyperparameter tuning, I determined that the ideal number of neighbors for the K Nearest Neighbor model was 1, meaning it had the highest accuracy when only compared to 1 close match. This makes sense since the ragams are very distinct, so there likely wouldn't be too many close matches– very few would be considered a "match". With too many neighbors, these matches get farther from representing the features present, making the model less accurate. The K Nearest Neighbor model was also not ideal for many reasons. Firstly, it's a simpler model compared to

Decision Tree and Random Forest, so it doesn't consider many nuances that help identify the ragam. Additionally, a neighbor number of 1 is a red flag since it indicates that each category is extremely distinct. Therefore, categorizing the features based on other "matches" isn't ideal because there wouldn't be many closely related matches. Instead, many matches would be somewhat similar but still very different, which could cause the model to predict wrongly. Because of these factors, the KNN model had the lowest accuracy of 92.4%. Decision Tree was better because it's well suited for dealing with distinct categories. However, it naturally had a lower accuracy than the Random Forest; the Random Forest model consists of many decision trees that factor into the final prediction, so the Random Forest model has more opportunities to learn and comprehend the characteristics than a Decision Tree model. Therefore, the accuracy of the Random Forest was higher than that of the Decision Tree, despite both being high.

I was able to get over certain limitations by using the Random Forest model rather than the K Nearest Neighbors model. B.T. Rao, S. Chinnam, P.L. Kanth, and M. Gargi of The Science and Information Organization also used K Nearest Neighbors as their model. Their model did quite well, with its highest performance yielding an accuracy of around 96%. However, I was able to get an even higher accuracy by using Random Forest. As detailed in the paragraph above, due to the nature of the various ragams, the K Nearest Neighbors model doesn't work well in this situation, so its results are less accurate. By using a Random Forest model, the model was better able to learn and analyze the characteristics of each ragam, leading to better results. The final accuracy of my Random Forest model was 99.1%, which is higher than their K Nearest Neighbors model accuracy of 96% (Rao et al.).

## LIMITATIONS

However, my model still has some limitations. Firstly, the dataset I used to train the model does not include all the ragams. There are 72 melakarta ("parent") ragams total, and more than a thousand janya ("child") ragams. Researchers such as V. Kaimal and S. Barde of the Dept. MATS School of Information Technology incorporated all 72 ragams into their dataset so that their model would recognize any of these "parent" ragams (Kaimal and Barde), something mine cannot do. The dataset I used only includes 32 ragams total, with 7 melakarta ragams and 25 janya ragams. This certainly does not represent the full range of ragams in Carnatic music, so if the model was given features from a ragam that isn't present in the dataset, it would classify it incorrectly because it's only been trained on 32 ragams. Regarding janya ragams, there are simply too many for the model to be trained on. So, it's more practical to pick a popular few hundred that the model can use to train. This is a general limitation for any ragam-identifier model; it cannot plausibly identify all existing janya ragams.

Another limitation is that the model does not directly extract the features from the audio. Rather, it requires all the features to be mapped out and inputted before it can be analyzed. If any data were to be given to the model to classify its ragam (outside of the dataset or being preprocessed in the same manner) it would need to be the data of the various audio features. The model won't be able to predict the ragam if only given the raw audio. This could become a potential issue if I were to apply this model to an application for users to input audio for the model to classify its ragam. The user would only be expected to input raw data, but this model cannot process the raw data itself. Another program would be needed to extract the features from the raw audio and organize it into data tables, which could then be inputted into the model. This situation can be constructed; however, the model itself cannot extract the features, which is one of its limitations.

This limitation is something that I wasn't able to address but has been addressed by other researchers. Many people have built models that use audio frequencies to classify the ragam, just like the model I constructed. However, they also have systems in their model to extract the frequencies and other features from the audio to analyze them. So, raw audio could be inputted into these models. For example, R. Sridhar and T.V. Geetha of Anna University segmented the audio into shorter pieces and used the model to analyze each piece for swaras and frequency patterns that matched certain ragams (Sridhar and Geetha). The extraction of the features was part of their methodology, while my model can only analyze features that are already extracted and organized.

Lastly, a limitation of the model is that it would not be able to classify an audio accurately if it contained multiple ragams. Certain songs in Carnatic music contain different ragams for different sections. The songs are known as Ragamalika, which means "a garland of ragams." The name highlights their quality of combining multiple ragams into a single composition. The model is

only trained to recognize one ragam from the features of the audio, so it would only output one ragam as its prediction. It's not trained to recognize different ragams within one clip, so it won't consider the scenario of multiple ragams in a single song. Therefore, Ragamalika compositions would always be incorrectly identified, since the model would either predict only one of the ragams present or predict an incorrect ragam entirely.

## CONCLUSION

Carnatic ragams are incredibly complex and consist of many subtle nuances, making them difficult to identify by ear. However, recognizing unique features like frequency patterns can help machine-learning models classify these ragams more accurately. I developed a model using a Random Forest Classifier trained on audio features such as spectral bandwidth and MFCC coefficients, which can predict the Carnatic ragam of an audio clip with 99% accuracy. The model was trained on a dataset from Kaggle containing 1,112 records of 30-second audio files along with their audio features. The dataset contains information on 32 ragams, including 7 melakarta "parent" ragams and 25 janya "child" ragams. The model's basic disadvantage, however, is its inability to classify the ragam based on the raw audio alone. The audio features must be extracted by a separate system before being input into the model, as the model itself cannot process the raw audio. Nevertheless, this model provides a way to identify ragams based on audio features without needing expert knowledge, making Carnatic music more accessible to people who aren't formally trained. Specifically, the model can help people enjoy the subtle nuances of each ragam while listening. It also serves as a useful tool for students learning Carnatic music, helping them learn to distinguish between different ragams. To make this model even more practical, it could be developed into a mobile app, allowing users to easily identify ragams in their daily lives. This would make it a valuable resource for both Carnatic music students and anyone who enjoys listening to the art form.

## References


Kaimal, Veena, and Snehlata Barde. “Introduction to Identification of Raga in Carnatic Music and Its Corresponding Hindustani Music.” *International Journal of Computer Sciences and Engineering*, ISROSET, 30 June 2018, https://www.researchgate.net/profile/Veena-Kaimal-2/publication/326554619_Introduction_to_Identification_of_Raga_in_Carnatic_Music_and_its_Corresponding_Hindustani_Music/links/6230d429069a350c8b8ff4a7/Introduction-to-Identification-of-Raga-in-Carnatic-Music-and-its-Corresponding-Hindustani-Music.pdf. Accessed 2024.

Natesan, Sanjay, and Homayoon Beigi. “Carnatic Raga Identification System Using Rigorous Time-Delay Neural Network.” *arXiv.Org*, 25 May 2024, arxiv.org/abs/2405.16000. Accessed 2024.

Rao, B Tarakeswara, et al. “Automatic Melakarta Raaga Identification Syste: Carnatic Music.” *International Journal of Advanced Research in Artificial Intelligence (IJARAI)*, The Science and Information (SAI) Organization Limited, 10 Dec. 2012, thesai.org/Publications/ViewPaper?Volume=1&Issue=4&Code=IJARAI&SerialNo=6. Accessed 2024.

Sridhar, Rajeswari, and T. V. Geetha. “Raga Identification of Carnatic Music for Music Information Retrieval .” *International Journal of Recent Trends in Engineering*, May 2009, https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=a389a0e91e1e587e384c1e54d849f69aa8055c5e. Accessed 2024.

Satish68l, Sanjana. “Ragas with Features in Indian Classical Music.” Kaggle, 30 Aug. 2023, www.kaggle.com/datasets/sanjanasatish68l/carnatic-ragas-with-features?resource=download. Accessed 2024.