Machine Learning Approach Towards Raga Identification

A Dissertation for

Course code and Course Title: ELE-625 Project

Credits: 16

Submitted in partial fulfillment of Master's Degree
M.Sc. in Electronics
by

MR. NAVIN D MATONKAR

Seat No.: 22P0360012

ABC ID: 996549110353

PRN:201805432

Under the Supervision of

DR. MARLON SEQUIERA

School of Physical and Applied Sciences
Electronics

GOA UNIVERSITY
MAY 2024

Examined by:

Seal of the School

DECLARATION BY STUDENT

I hereby declare that the data presented in this Dissertation report entitled, **Machine Learning Approach Towards Raga Identification** is based on the results of investigations carried out by me in the M.Sc Electronics at the School of Physical and Applied Sciences, Goa University under the Supervision of Dr. Marlon Sequiera and the same has not been submitted elsewhere for the award of a degree or diploma by me. Further, I understand that Goa University or its authorities will be not responsible for the correctness of observations / experimental or other findings given the dissertation. I hereby authorize the University authorities to upload this dissertation on the dissertation repository or anywhere else as the UGC regulations demand and make it available to any one as needed.

Mr. Navin D. Matonkar
Seat no: 22P0360012
Electronics Discipline
School of Physical & Applied Sciences

Date: 14/05/2024

Place: Goa University

COMPLETION CERTIFICATE

This is to certify that the dissertation report **Machine Learning Approach Towards Raga Identification** is a bonafide work carried out by Mr Navin D. Matonkar under my supervision in partial fulfillment of the requirements for the award of the degree of Masters in the Discipline Electronics at the School of Physical and Applied Sciences, Goa University.

Dr. Marlon Sequiera

Electronics Discipline

School of Physical and Applied Sciences

Date: 13/05/2024

Prof Ramesh V. Pai

School of Physical and Applied Sciences

Date:

Place: Goa University

School of Physical and Applied Sciences Goa University

School/Dept Stamp

Table Of Content

Preface

This dissertation represents the culmination of several months of research and hard work. It is with great pleasure that I present it to the academic community. Throughout this journey, I have been fortunate to receive support and guidance from many individuals and the institution, to whom I owe immense gratitude. The dissertation is organized as follows: Chapter 1 provides an introduction to the topic, including its significance and relevance. Chapter 2 reviews the existing literature on Raaga Identification using different methods, synthesizing key findings and identifying gaps in the literature. Chapter 3 outlines the methodology employed in this study, detailing the research design, data acquiring methods, and analytical approach. Finally, Chapter 4 presents the empirical findings and analysis.

Acknowledgements

This dissertation would not have been completed without the help and guidance of several individuals who in one way or the other contributed and extended their valuable assistance in the preparation and completion of this study, it is a pleasure to thank those who made it possible.

I would like to express my gratitude to the faculty of the Electronics Department of Goa University for giving me this opportunity to learn throughout this process of doing my project.

The completion of this project could not have been possible without the expertise of Dr. Marlon Sequiera, our project guide.

I would like to thank the teaching faculty, Prof. Rajendra S. Gad, Dr. Jivan S. Parab, Dr. Narayan Vetrekar , Dr. Sandeep Gawali and Dr. Aniketh Gaonkar for their support throughout the completion of this project.

I would also like to thank Mr. Vishant Malik (Lab Technician) for his assistance and provision of necessary components.

A sincere thank you to Mrs. Ashwini and Sir Ramchandra for their help and support.

Last but not the least, I would like to thank my friends and family for always extending their helping hand and support.

Tables and figures

Abbreviation Used

MFCC	Mel Frequency Cepstral Coefficient
ML	Machine Learning
SVM	Support Vector Machine
CNN	Convolutional Neural Network
RF	Random Forest
ICM	Indian Classical Music
Sa	Shadja
Re	Rishabha
Ga	Gandhara
Ma	Madhyama
Pa	Pancham
Dha	Dhaivata
Ni	Nishad

Machine Learning Approach Towards Raga Identification

Navin D. Matonkar

MAY, 2024

Abstract:

Raga is the essence of Indian Classical Music that is used in its composition, performance, improvisation and organization. This work has developed a method for Raga identification using MFCC with machine learning algorithms and deep learning . A comparative study is done on the efficiency of the different machine learning model for Raaga Identification. Raaga Identification helps to Identify different Raaga. A comparative study is done on the efficiency of the different machine learning model for Raaga Identification. Raaga Identification helps to Identify different Raaga. Automatic identification of the underlying raga in an Indian Classical song has applications in areas like music recommendation and indexing. The achieved accuracy are 94.99, 90, 80.5, 90 for CNN ,SVM ,XBG ,Random Forest respectively.

CHAPTER 1

1.Introduction

1.1 Background

What is Indian Classical Music?

India as a country is widely famous for its rich cultures, and traditions. Everything about India is custom opulent. From our attire to the cuisines, the languages to the holidays, and from our taste buds to our music. Indian Classical music is one of the oldest forms of art. It originated from the Vedic Literature of *Hinduism* and the ancient *Natyashastra*. Indian Classical Music can be divided into its two base elements which are *Raga* and *Tala*. The *raga* mainly forms the musical structure of the song while the *tala* measures the time. Even though the range of Indian Classical Music is so vast and it is an ancient art, it can be separated into two main parts geographically – The North Indian part which is called the Hindustani Music, and the South Indian Classical Music, which is termed as the Carnatic Music[1].

Hindustani Classical music is all about improvisation and playing and exploring all aspects of the *raga* and Carnatic Classical Music is all about *tala* or composition-based. But the ancient *Natya Shastra* text is at the root of Indian Classical Music in general. It is so diverse and authentic that with time it has adapted itself to various regional styles, dialects, and languages, take for example Bengali Classical Music. Before the ancient Delhi Sultanate, there wasn't much difference between Hindustani Classical Music and

Carnatic Classical Music. But after the Sultanate empire, North India was differentiated from South India and the traditions took different forms, adapting themself to their customs and folklores and vibes. That's how Carnatic and Hindustani Classical Music was originated.

The songs composed under Indian Classical Music are heavily based on our Customs, religious traditions, and holidays and hence the instruments used in the production of this music are kept pure and true to their name, by using traditional Indian instruments. Instruments like Table, Harmonium, Sitar, Sarod. The sounds produced by these Classical Instruments are some of the crispest sounds you'll ever hear.

Some of the famous Indian Classical Musicians are *Ustaad Bismillah Khan* – the famous *Shehnai* players who grasped the reign of the traditional *Shehnai* music like no other and no one else since then has been able to replace or match those standards. Another extremely popular artist was *Pandit Ravi Shankar*, globally known for his iconic contribution to Indian Classical Music, he was widely famous for his *Sitar* skills. *Hariprasad Chaurasia* is another great example of Indian Classical Musicians, he is known as the legendary flutist. This one is quite a heartbreaking one; the famous singer *Jagjit Singh* who sang the song "*Chitthi Na Koi Sandes*h" was also extensively trained in Indian Classical Music. Female artists like *Lata Mangeshkar*, *Asha Bhosle*, *Gauhar Jaan*, are amongst some of the most legendary Indian Classical Musicians of our country.

An interesting fact about Indian Classical Music is that it follows the "*Gharana Culture*". According to the *Gharana Culture*, one family practices one raga or instrument for

generations. A lot of the most famous Indian Classical Music Artists belonged to the *Gharana Culture*.

Hindustani Classical Music

The Indian Classical Music branches into the North Indian Classical Music or widely known as the Hindustani Classical Music or *Shashtriya Sangeet*. Originated in the 12th century, it has a 12-note scale that focuses on the improvisation and exploration of *ragas.*

There are three main octaves in this music form, mainly, low, medium, and high, and each octave resonates with a body part. Like a low octave in the heart, medium octave in the throat, and high octave in the head. Even though Hindustani music solely focuses on the vocals, in recent years and especially outside India and Asia, Instrumental Hindustani Classical Music is extremely popular. Partly because of the language barrier and partly because of the faster and refreshingly different instrumental sounds.

Examples of such classical compositions are innumerable such as "*Bansiya Bajawat"* based on Raga *Yaman, "Jaago Mohan Pyaare"* based on Raga *Bhairav*, *"Jaao Shyam Tum Humse Na Bolo"* based on Raga *Bhairvi.*

Carnatic Classical Music

The other branch of Indian Classical Music is Carnatic Music or *Karnataka Sangitam*. The difference between Hindustani Classical Music and Carnatic Classical Music comes with the geography of the states. Carnatic music is based around the South Indian Languages, widely famous in states like Tamil Nadu, Karnataka, Telangana, Kerala, Andhra Pradesh, and Kerala.

Another major difference is that Carnatic Music focuses more on the vocals. All the compositions, even the instrumental ones are supposed to be sung. The rules of Carnatic Classical Music are also more stringent than that of Hindustani Classical Music.

One of the best aspects of India is the way every culture and customs are celebrated here. Carnatic Classical Music is also widely celebrated throughout Southern India. The city of Chennai holds a six-week-long music season in which Carnatic Music is widely appreciated. The *Ragas* may be termed or named differently in Carnatic Classical Music than Hindustani Classical Music but their tone remains the same.

What is Thaats?

In Hindustani (North Indian) classical music system, a thaat represents a Parent Scale that uses at least 7 swars/notes(out of the 12 swars) to make a Raaga or Raag. Out of these seven notes–*sa re ga ma pa dha ni*–Sa and Pa remains constant without variations but other notes: re ga ma dha and ni each have two variants (natural vs. flat, or natural vs. sharp), so there is a possibility of getting 32 different *thaats*, but 10 thaats are commonly used in Hindustani music[2].

Pandit Vishnunarayan Bhatkhande:

The modern *thaat* system was created by Vishnu narayan Bhatkhande(1860–1936). He modelled his system after the Melakartha system of carnatic music classification.

A Thaat always has the following rules:

A thaat must have seven tones out of the twelve tones [seven natural, four flat (Re, Ga, Dha, Ni), one sharp (Ma)]

The tones must be in ascending sequence: Sa Re Ga Ma Pa Dha Ni

A thaat cannot contain both the natural and altered versions of a note

A thaat, unlike a raga, does not have separate ascending and descending lines

A thaat has no emotional quality (which ragas, by definition, do have)

Thaats are not sung but the ragas produced from the thaats are sung ,10 Thaats as follows with respect to the Raga :

1. Kalyaan Thaat – This Thaat has a group of evening ragas. As this Thaat is considered as a blessing-seeking and soothing. it is sung at the beginning of a concert in the evening. Like Bhairav, this Thaat too is vast and so has many variations like Shuddha Kalyan, Shyam Kalyan, Yaman Kalyan, Anandi Kalyan, Khem Kalyan (Haunsdhwani + Yaman), Savani Kalyan etc.

2. Bilaawal Thaat - It's the most fundamental of all the ten Thaats and is related to the morning.

3. Khamaaj Thaat – It is obtained by replacing the Shuddha Nishad of Bilawal by Komal Nishad. Its nature is romantic (Shringar Ras).

4. Bhairav Thaat – Bhairav Thaat uses Komal Rishabh and Komal Dhaivat. It has manly and austere feelings. Itf is exceptionally huge and so constitutes a large number of note combinations like Ahir Bhairav, Alam Bhairav, Anand Bhairav, Bairagi Bhairav, Beehad Bhairav, Bhavmat Bhairav, Devata Bhairav, Gauri Bhairav, Nat Bhairav, Shivmat Bhairav etc. It is typically sung in a devotional mood in the early morning.

5. Poorvi Thaat – This is the mixture of komal Dhaivat to Marwa Thaat. It is intensely sober and is sung at the sunset.

6. Maarva Thaat – It's a combination of komal Rishabh and Kalyan Thaat. This Thaat coveys the mood of the sunset and so has a feeling of nervousness.

7. Kaafi Thaat – Kafi Thaat uses the Komal Gandhar and Komal Nishad. It's a late evening raga and is associated with the spring.

8. Aasaavari Thaat – Asavari Thaat is a blend of Komal Dhaivat and Kafi Thaat. It has the nature of renunciation and sacrifice as well as suffering. It is apt for late morning.

9. Bhairavi Thaat – Bhairavi uses all the komal swars, Rishabh, Gandhar, Dhaivat, Nishad. Bhairavi Raga is named after the cosmic feminine power (i.e. Shakti or Maa Durga). It conveys the feeling of devotion and compassion. It is actually sung in the early morning, yet customarily its singing ends the program.

10. Todi Thaat – This is regarded as the king of all Thaats, as symbolizes pleased worship with a placid, warm outlook and is sung in the late morning.

The specifications of each of the above thaats and popular Raagas in each Thaat are given below:

1. Kalyaan Thaat : S R G Ḿ P D N (Teevra Madhyam Ḿ)

Raagas: Kalyaan/ Shuddha Kalyaan, Shyam Kalyan, Bhupaali, Chaandni Kedaar, Chhayaanat, Gaud Saarang, Hameer, Hindol, Kaamod, Kedaar, Nand, Yaman/ Aiman, Yaman Kalyaan

2. Bilaawal Thaat : S R G M P D N (All Shuddha Swars)

Raagas: Bilaawal/ Alahiya Bilaawal, Bihaag, Nat Bihaag, Bihaagada, Deshkaar/ Desikaar, Durga, Hamsadhwani, Hemkalyaan, Kaushik Dhwani, Shankara

3. Khamaaj Thaat : S R G M P D n (Only Nishaad n is komal)

Raagas: Khamaaj, Desh, Gara, Gaud Malhaar, Gaawati, Gorakh Kalyaan, Jayjayvanti, Jhinjhoti, Jog, Kalaavati, Rageshri, Saraswati, Sorat, Tilak Kaamod, Tilang

4. Bhairav Thaat : S r G M P d N (Rishabh r and Dhaivat d are komal)

Raagas: Bhairav, Ahir-Bhairav, Nat-Bhairav, Gauri, Gunkali, Jogia, Kaalingadaa, Ramkali, Vibhaas

5. Poorvi Thaat : S r G Ḿ P d N (Rishabh r, Dhaivat d, both are komal; and Madhyam Ḿ is Teevra)

Raagas: Poorvi, Basant, Lakshmi Kalyaan, Lalit, Paraj, Puriyaa Dhanashri, Shree

6. Maarva Thaat : S r G Ḿ P D N (Rishabh r is komal, Madhyam Ḿ is Teevra)

Raagas: Maarwa, Bhankar, Bhatiyaar, Lalit, Puriyaa, Puriyaa Kalyaan, Sohini, Vibhaas

Kaafi Thaat : S R g M P D n (Gaandhaar g, Nishaad n, both are komal)

Raagas: Kaafi, Abhogi, Bageshri, Bahaar, Barwaa, Bhimpalaasi, Brindaavani Saarang, Chandrakauns, Dhaani, Jog, Megh, Ramdaasi Malhaar, Surdaasi Malhaar, Mian Malhar, Nayaki Kaanhadaa, Patdeep/ Patdeepki/ Pradipki, Piloo, Shahana, Shuddha Saarang

8. Aasaavari Thaat : S R g M P d n (Gaandhaar g, Dhaivat d, and Nishaad n, all three are komal)

Raagas: Aasaavari, Adaanaa, Darbaari Kaanhadaa, Kaunsi Kaanhadaa, Desi, Dev Gandhaar, Jaunpuri

9. Bhairavi Thaat : S r g M P d n (Rishabh r, Gaandhaar g, Dhaivat d, and Nishaad n, all four are komal)

Raagas: Bhairavi, Bhupal Todi, Bilaskhaani Todi, Komal Rishabh Aasaavari, Maalkauns

10. Todi Thaat : S r g Ḿ P d N (Rishabh r, Gaandhaar g, and Dhaivat d, all three are komal; and Madhyam Ḿ is Teevra)

Raagas: Todi, Gurjari Todi, Lilaavati, Madhuvanti, Multaani

What is a Raaga?

The notion of a Raga is at the foundation of Indian Classical Music. Simply put, a Raga uniquely defines a set of musical notes and their allowed arrangements to form a melody to evoke a certain mood.

In Sanskrit, a Raga means “something that colors your mind.” Within Indian classical musical systems, a Raga has the power to create very specific emotions in one’s mind. A range of emotions such as joy, sadness, happiness, romance, yearning, devotion, and more can be expressed through Raags. Some Raags are seasonal; they enhance the listener's mood through association with a particular season, such as spring or monsoon.

Ancient scriptures define a Raga as a composition of sounds capable of bringing joy to the human heart while attaining beauty through specific movements of notes and phrases[3].

A raga gets its unique identity by a variety of characteristics which are briefly described below –

1. Svaras - The seven solfege symbols Sa, Re, Ga, Ma, Pa, Dha, Ni used in ICM are known as svaras. All svaras, except Sa and Pa, have two to three variations. Every raga typically consists of a set of five to seven svaras. Each svara has well defined functional role in the context of a given raga . Unlike western music, Svaras don't have a fixed pitch value, rather they are relative to each other.

2. Vadi and Samvadi - Vadi and Samvadi are the most significant and second most significant svara in a raga respectively. Significant svara usually refers to the svara which is repeated the most or the svara on which the artist can pause for a significant amount of time.

3. Tonic Pitch - It is the base pitch of the performer which acts as a reference throughout the performance. The artist is free to choose the Tonic Pitch which corresponds to the svara Sa and all the other svaras derive their values relative to it. The concept of tonic is fundamental to any raga in ICM . It should be noted that Tonic Pitch is a pitch value, not a pitch class.

4. Arohana-Avrohana - The ascending and descending progression of svaras in a raga is known as Arohana and Avrohana respectively.

5. Gamaka - Gamakas are the various improvisations and ornamentations that can be brought in by the artist while performing. There are around 15 different gamakas in

Indian Carnatic Music . For example, kampitam is a gamaka characterized by an oscillatory pitch movement around the svara.

6. Raga Motif - It is a combination of melodic phrases that characterize a raga . It is the most noticeable feature used by humans to identify a raga. It is the building block of ragas on the top of which an artist can improvise.

7. Chalan - The melodic transition from one svara to another is different for every raga and is defined by the chalan of that raga. It also defines the amount of time an artist needs to spend on each svara .

What are the ingredients of a Raaga?

Every Raga has its own rules and features. Examples of these rules are:

•Only specific notes are allowed in the raga.

•The raga defines its most important notes, known as Vadi and Samvadi.

•It defines the key phrases that should be used often during improvisation.

•Special treatment must be given to some notes. ex. specific notes that must be sung with a glide(meend), an oscillation of a particular note(aandolan).

•The raga also sets its mood and tempo. Watch Bhairav videos for a demonstration of an inherently slow and a peaceful raga vs Deshkar, which has quick-moving notes.

Knowing the rules is necessary for an artist to practice the Raga accurately so that when it’s performed, the focus can be on the aesthetics. Knowing the rules of a raga, although

not required by a casual listener, will certainly make their listening experience more enjoyable.

How many Ragas are known today?

Indian Classical music is a very ancient art form. It has transformed over centuries due to various influences. And in every generation scholars have made invaluable contributions in analyzing and documenting the system of Raags. India's classical music system was known to have 4,840 Raags at some point in time. This art form is challenging to describe in a textual format; it is essentially an oral tradition. This music was best passed on down through the generations by Gurus to their disciples. Due to the challenges faced in documenting this system of music, many Raags were lost forever, and only a few hundred remain today.

What is Svara?

Svara (Sanskrit: स्वर *svara*) is a word that connotes simultaneously a breath, a vowel, the sound of a musical note corresponding to its name, and the successive steps of the octave or *saptaka*. More comprehensively, it is the ancient Indian concept about the complete dimension of musical pitch.Most of the time a *svara* is identified as both musical note and tone, but a tone is a precise substitute for sur, related to tunefulness. Traditionally, Indians have just seven *svara*s/notes with short names, e.g. saa, re/ri, ga, ma, pa, dha, ni which Indian musicians collectively designate as *saptak* or *saptaka*. It is one of the reasons why *svara* is considered a symbolic expression for the number seven[4].

Svara (long)	Sadja(षड्ज)	Rishabha (ऋषभ)	Gandhara (गान्धार)	Madhyama (मध्यम)	Pañcham (पञ्चम)	Dhaivata (धैवत)	Nishada (निषाद)
Svara (short)	Sa (सा)	Re (रे)	Ga (ग)	Ma (म)	Pa (प)	Dha (ध)	Ni (नि)
12 Varieties (names)	C (sadja)	D ♭ (komal re), D (*suddha re*)	E ♭ (komal ga), E (*suddha ga*)	F (suddha ma), F♯ (*tivra ma*)	G (pancama)	A ♭ (komal dha), A (*suddha dha*)	B ♭ (komal ni), B (*suddha ni*)

Table 1: Svara in North Indian system of Rāga

Note Name	Notation ID	Solfa Syllable	Western Equivalent	Full Name
sa	S	sa	Unison	shadja
komal re	r	re	Minor second	komal rishabha
shuddha re	R	re	Major second	shuddha rishabha
komal ga	g	ga	Minor third	komal gāndhāra
shuddha ga	G	ga	Major third	shuddha gāndhāra
shuddha ma	m	ma	Perfect fourth	shuddha madhyama
tivra ma	M	ma	Augmented fourth	tivra madhyama
pa	P	pa	Perfect fifth	panchama
komal dha	d	dha	Minor sixth	komal dhaivata
shuddha dha	D	dha	Major sixth	shuddha dhaivata
komal ni	n	ni	Minor seventh	komal nishāda
shuddha ni	N	ni	Major seventh	shuddha nishāda
sa	S'	sa	Octave	shadja

Table 2; Svara Notation

Svaras	Frequencies
Sa	256
Re	280
Ga	312
Ma	346
Pa	384
Dha	426
Ni	480

Table 3 : Svaras with respect to their frequencies

How are Listeners Identifying Raagas?

Extensive Internalization Through Deliberate Practice:

Prolonged Sadhana: Singers of Hindustani music practice long periods of dedicated practice, often called sadhana. This practice emphasizes careful consideration of the subtle qualities of the various ragas. Singers carefully study the ascending and descending patterns (arohana-avarohana), the tonal center (vadi-samvadi) and each decorative flourish (gamaka) of each raga.

Mnemonics and Visualization Tools: Hindustani music teachers (gurus) often use mnemonics or visualization techniques to help singers remember the unique melodic structure of each Raga. These techniques can include associating the Raga with specific syllables for better memorization, creating mental images to represent the melodic flow of the Raga, or using spatial metaphors to describe the movement of notes within the Raga.

1.2 Aim and Objective

Machine learning approach towards Raga Identification

Objective

The primary goal of machine learning-based raga identification is to automatically categorize an Indian classical music composition into the appropriate raga. A model can be trained on a dataset of labeled raga recordings to accomplish this. The advantages are broken down as follows:

Automated Classification: Manually identifying ragas needs expertise because to their complexity and intricacy. This procedure is automated by machine learning, which makes it quicker and easier to use.

Raga identification serves as a foundational element for music information retrieval systems. These algorithms may detect a raga and utilize that information to find related music, classify music according to emotion or mood, or even create playlists depending on user preferences.

Education and Learning: Educational resources that teach students how to recognise ragas can be made using machine learning algorithms. They can also be used to evaluate raga performances and give artists constructive criticism.

1.3: Hypotheses

Hypotheses for machine learning approaches to raga identification in Indian classical music:

Feature-based approaches:

Melodic patterns: Ragas are defined by specific melodic movements and phrases. The idea is that the model may be trained to distinguish between different ragas by taking information from the audio, such as pitch contours, intervals, and melodic patterns.

Timbral characteristics:The instruments and techniques used to play different ragas give them unique tones. It is hypothesised that characteristics such as envelope forms, inharmonicity, and spectral qualities can be utilised for raga identification.

Rhythmic features:Ragas differ in terms of pace and rhythmic patterns. It is hypothesised that features that can be extracted to help with classification include pace, rhythmic patterns, and the existence of particular rhythmic elements.

Combining features:Using a combination of melodic, timbral, and rhythmic characteristics will probably result in a more accurate raga identification. A model trained on these combined features should perform better than a model trained on separate features, according to the hypothesis.

Deep learning approaches:

Learning complex relationships: It is possible for deep learning algorithms to automatically discover intricate connections between raga labels and audio data. Deep neural networks are thought to be able to pick up on minute details in audio recordings that are hard to pick up on with manually created features[4].

Transfer learning: Raga recognition can be improved by fine-tuning pre-trained deep learning models on sizable music datasets. It is hypothesised that these models can increase the accuracy of raga classification by utilising their acquired knowledge of music representation.

Multimodal learning: Additional information like as performance context or raga descriptions may be useful for raga identification. It is hypothesised that adding this multimodal data will enhance the model's raga identification capabilities.

1.4: Scope

When it comes to identifying ragas in Indian classical music, machine learning (ML) presents a number of promising advantages.

Automated Raga Classification: Traditionally, brilliant musicians are needed to identify ragas. This procedure is automated by ML, making it, Faster: Automatically analyse large collections of music and instantly recognize ragas for use in music recommendation systems, among other uses. Accessible:Quickly analyse vast collections of music and instantly recognize ragas for use in music recommendation systems, among other uses.

Even more Knowledgeable Music Information Retrieval (MIR): Raga identification is the bedrock of MIR systems.

Categorization and Search: Look for similar tracks according to raga and sort music collections by moods expressed through ragas.Smart Playlists: Create playlists that match user preference or time of day with raga features.Education And Learning Tools: ML can be particularly useful in music education: Applications For Interactive Learning: Make interactive applications that will help students learn how to recognize ragas through gaming or live feedback.Analysis Of Performance: Study the performance of ragas and provide data-driven advice for improvement on musical technique.More Than Classification: Though classification may be one goal among many, ML has greater potential: Raga Derivation And Exploration: Generate new ragas or their variations from existing ones using generative models. Assistance In Music Composition: Invent AI tools that recommend melodic phrases or rhythmic patterns specific to a chosen raga, thus aiding composition.

Chapter 2

2. Literature Review

Melody is the spirit of Indian classical music for raga identification.

In 2018, Makarand Velankar et al. evaluated several raga recognition systems using a dataset of monophonic vocal audio samples ranging from 30 to 180 seconds. The PCD technique achieved an accuracy of 66.66% for 30-second samples, while the N-gram approach achieved a 58.33% accuracy in pattern identification using pakad or conspicuous patterns of raga. N-grams are relevant for distinguishing ragas because they provide significant data about tune structure, which is maintained during performance and creation. This approach relies on the succession of 'n' notes, with a 2-gram addressing a two-note succession and a 3-gram addressing a three-note succession. The tune structure separates two ragas that may be otherwise very similar[5].

Analyzing Acoustics of Indian Music Audio Signal Using Timbre and Pitch Features for Raga Identification.

In 2019, Waghmare and Sonkamble proposed a classification system for Indian classical music based on raga, aimed at helping e-students, authors, and artists. The system uses different feature extraction algorithms, such as MFCC, LPCC, and PCP, for music examination and characterization. MFCC feature extraction describes the power spectrum envelope of a single frame in the MFCC feature vector. The system also uses three pitch detection algorithms, namely, autocorrelation, cepstrum, and harmonic product spectrum, to determine the pitch values in an input. The average accuracy of MFCC features is

above 85%, and the accuracy using pitch features is greater than 90% in all three approaches[6].

RAAGANG—A Proposed Model of Tutoring System for Novice Learners of Hindustani Classical Music

In 2020 Kunjal Gajjar et.al proposed a model for a teaching system that hears the singer's voice and assesses it using raga rules/convention. Raga and Taal are the two most crucial components of composition in Hindustani music, Hindustani classical music follows the "Guru Shishya Padhati" style of learning.The System has three sub-modules.Note Identification, Note Evaluation and Error Correction. Context Sensitive Error Correction is responsible for replacing the error note there by recommending the proper note.This module determines the context of the error note and offers suggestions based on the past and future note sequences. accuracy: Note Identification module is tested comparing machine generated notes with experts identified notes 89.89%,Note evaluation 180 compositions for 8 ragas 100%error identification is archived , scale of accuracy 56.32% of suggested correction and as exactly original composition ,70.96% suggested corrections are excellently adequate and 25.08% are fairly adequate as per aesthetics correctness scale[7]

Tansen : A System For Automatic Raga Identification

In their paper, Pandey et al. proposed the TANSEN system for raga recognition, which uses note duration heuristics and hill peak duration for note transcription. They also employed Pakkad matching and Hidden Markov Model for plain and improvised raga

identification. The system achieved an accuracy of 77% for plain raga identification and 87% for raga identification with Pakad matching[8]

Raga Identification Using Convolutional Neural Network

In 2019, Ankit Anand proposed a CNN based model to extract melodic features from the predominant pitch values of a song. CNN for raga identification ,Input layer- Songs as Images (Pitch Variation over time), Convolutional Layer , Pooling , Fully Connected Layers and Softmax where the model achieved an accuracy of 96.7% and 85.6% respectively[9].

Machine Learning For Raga Identification In Indian Classical Music

In 2019 PRANATI CHINTHAPENTA, Proposed outlines a comprehensive approach for automated raga identification in Indian classical music using machine learning techniques. The methodology involves data collection, feature extraction, model selection, and evaluation metrics. The proposed approach aims to leverage a curated dataset of Indian classical music recordings, annotated with raga labels, and extract relevant features such as pitch, timbre, and rhythm patterns. The model selection involves traditional models such as KNN and SVM, as well as neural networks like LSTM and CNN. The proposed approach aims to contribute to the field of music information retrieval and enhance model robustness and accuracy through future research with ensemble methods, transfer learning, and larger datasets . The Train accuracy is 87% and the Test accuracy is 30%[10].

Raga Identification Using MFCC and Chroma features

In 2017, Kavita.M Deshmukh, P. Deore proposed a idea of Raga Identification Using MFCC and Chroma features This work attempts to solve the raga classification problem in a non-linear SVM (support vector machine) framework using a combination of two relevant features that represent the similarities of a music signal using two different features MFCC (Mel Frequency Cepstral Coefficient) and Chromagram, accuracy rate is 93%[11].

Machine Learning Based Indian raga Identification for Music Therapy

In 2022, Kiran proposed a idea for Machine Learning Based Indian raga Identification for Music Therapy using DWT Co-efficient Computation, MFCC Computation .. In Indian classical music, ragas are utilized to symbolize different moods. Furthermore, a certain Raga may intensify a certain feeling. Accuracy is 90, 96% [12].

Raga Identification Based on Normalized Note Histogram Features

In 2015, R. Pradeep et. al proposed a discriminative method to identify raga of a polyphonic music clip using Normalized Note Histogram (NNH) features.A salience-based method is used to extract the pitch value sequence from the polyphonic music signal.Note-sequence is obtained from the pitch values by applying the tonic frequency value.110 clips from a music database are used to validate the suggested classifier and normalized histogram features. 82.34% is the observed accuracy of the proposed classifier[13].

Towards Raga Identification of Hindustani Classical Music

In 2019,Jyoti A. Lele et.al proposed a raga identification system with different methods like Fourier transform, chromogram and spectrogram with a database of few santoor samples and few songs in light music to get reasonable accuracy for the raga Identification ,accuracy was good for santoor samples upto 75% [14].

Raga classification using enhanced spatial bound whale optimization algorithm

In 2023,B.S.Gowrishankar,et.al proposed enhanced spatial bound whale optimization algorithm (ESBWOA) is used that overcome the feature selection problem of high dimensional features. In addition to this, a weighted salp swarm algorithm (SSA) is used for selecting the tone-based features from the ragas based on amplitude or each raga sample[15]

Indian Classical Raga Identification using Machine Learning

In 2021,Dipti Joshi,et.al proposed classification of different ragas like Yaman and Bhairavi by applying K-Nearest-Neighbor (KNN), Support vector machine (SVM) machine learning algorithms. Automatic raag classification has been discussed and attempted previously. Various methods have been used like pitch class distribution (PCD), HMM, finite automata modelling of the musical rules, pitch class string formation[16].

Carnatic Music Identification of Melakarta Ragas through Machine and Deep Learning using Audio Signal Processing

In 2023,Kshitiz Kumar,et.al proposed machine learning and deep learning models to predict the raga of Indian classical music from a given .wav file using the Librosa library, a widely used Python package for music analysis .study achieved 98.98% testing accuracy on a subset of 10 ragas using raw spectrograms and deep learning models[17].

Raga classification using enhanced spatial bound whale optimization algorithm

In 2023,B.S.Gowrishankar,et.al proposed audio features such as mel frequency cepstrum coefficients (MFCCs), spectral flux, short time energy, audio feature extractor, and spectral centroid features are used for the prediction of a raga. Enhanced spatial bound whale optimization algorithm (ESBWOA) , Weighted salp swarm algorithm (SSA) for feature selection ESBWOA accuracy 94.91%, SVM 70.52% , CNN 94%[18]

Deep Learning-Based Classification of Indian Classical Music Based on Raga

In 2023 ,Singha, Anupam, et al. a convolutional neural network was proposed for raga classification, which takes the spectrograms of the audio note and identifies the raga of the note. proposed convolutional neural network model achieved a precision of 97.9% on raga classification[19]

Raga Recognition Using Neural Networks and N-Grams of Melodies

In 2023,Sharma, Ashish, and Ambuja Salgaonkar proposed a neural network-based classifier for the recognition of the six ragas: Alhaiya Bilaval, Bhairav, Bhimpalasi,

Vrindavani Sarang, Kedar, and Yaman, from Hindustani classical music has been constructed.Neural network-based classifier for raga recognition ,N-grams of melodies used as input for the system.Classifier with 94% accuracy in recognizing six ragas[20].

Machine Learning Based Indian Raga Identification for Music Therapy

In 2022, ,K, Anitha,et.al a machine learning-based algorithm is proposed to identify the Raga recognition for music therapy, which uses MFCC (Mel Frequency Cepstral Coefficients) feature along with pitch and chroma information for feature extraction 92.34% accuracy using MFCC, pitch, and chroma features with KNN classification[21]

A Deep Learning Based Approach for Janya Raga Classification in Carnatic Music

In 2022,Kavitha, P., et al. proposed research aims to investigate the effectiveness of deep learning models for janya raga classification in Carnatic music. The research question we seek to address is: "How effective are deep learning models for janya raga classification?" Our hypothesis is that deep learning architectures can effectively capture intricate melodic nuances and outperform traditional methods. To test our hypothesis, we will curate a dataset of Carnatic music recordings containing various janya ragas and preprocess the audio data by extracting relevant features such as Mel-frequency cepstral coefficients and chroma features. We will then design a deep learning architecture tailored for raga classification, considering CNNs or RNNs with attention mechanisms. We will experiment with different layer configurations and activation functions to optimize the model's performance. To evaluate performance, we will split the dataset into training, validation, and test sets, train the model using stochastic gradient descent or

Adam optimizer, and assess the model's accuracy, precision, recall, and F1-score. We will also perform cross-validation to assess the model's generalization capabilities. 1D CNN-LSTM model achieved 82.0% accuracy[22]

A Comparison of Audio Preprocessing Techniques and Deep Learning Algorithms for Raga

In 2022, Hebbar. et.al present study aims to examine the significance of raga classification in Indian classical music and the role of audio pre-processing techniques and deep learning algorithms in achieving accurate results. The data collection process will involve the utilization of audio recordings of ragas, The extracted features will include Mel-frequency cepstral coefficients and chroma features. The deep learning architectures explored will include convolutional neural networks and recurrent neural networks. The models will be trained through cross-validation and hyperparameter tuning methods, and their accuracy will be evaluated through metrics such as accuracy and F1-score. The study will further examine the efficacy of pitch co-occurrence matrices and other computational approaches for raga identification, including spectral feature extraction and Hidden Markov Models. The proposed supervised learning model for raga identification in Carnatic music will also be analyzed, which involves training a machine learning model with features extracted from different Ragas and using this trained model to classify the Raga of a given audio signal. various deep learning models. achieving a testing accuracy of 98.1% [22]

Raga Recognition in Indian Carnatic Music Using Convolutional Neural Networks

In 2022,Rajan.et.al proposes a CNN-based sliding window analysis on mel-spectrogram and modgdgram for raga recognition in Carnatic music, which neither requires pitch extraction nor metadata for the estimation of raga.CNN-based sliding window analysis on mel-spectrogram and modgdgram enables raga recognition in Carnatic music without requiring pitch extraction or metadata, showcasing promising results for machine learning-based raga identification [23]

Raga Classification Based on Novel Method of Pitch Co-Occurrence

In 2022,V Rajadnya,et.al. proposed Raga identification in machine learning is achieved through pitch co-occurrence patterns extracted from audio recordings, aiding in efficient raga classification using K Nearest Neighbour (KNN) algorithm. This research has utilised the pattern developed due to co-occurrence of pitches of swaras for classification and K Nearest Neighbour (KNN) has been used as the classifier.Achieved accuracy: **90%** (cross-validation)[24].

A Survey on Computational Approaches for Raga Identification

In 2022, Patil, Surekha, et al. introduced a proposal for intricate melodic structures in Indian classical music that evoke distinct emotions and cultural contexts. In the field of Music Information Retrieval (MIR) systems, automatic raga identification holds significant importance for preserving India's rich musical heritage. With this aim, various computational approaches have been proposed for raga identification, which are discussed in this literature review. Among the proposed ideas, pitch co-occurrence

matrices have gained prominence in recent research. Deep learning paradigms such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) have been utilized to learn hierarchical representations from spectrograms or pitch contours. Methodologies such as spectral features extraction, including pitch histograms, chroma vectors, and timbral descriptors from audio recordings, and Hidden Markov Models (HMMs) for modeling the sequential nature of pitch sequences, have been employed for raga identification. The achieved accuracy levels reported for each of these methods hold significance in the academic domain[25].

Implementation of a Supervised Learning Model for Raga Identification in Carnatic Music

In 2021, Kaimal ,et.al. approach involves training a machine learning model with features extracted from different Ragas and using this trained model to classify the Raga of a given audio signal. Various methodologies have been proposed for this task, such as using the jump sequence of svaras or raw spectrograms from the audio source for classification. Additionally, an adaptive classifier based on Neural Networks (NN) has been suggested, whereby a feature set is used for learning. In one study, an accuracy of 99.56% and 99.43% was achieved for a set of ten and twenty Ragas, respectively, using techniques such as Artificial Neural Networks (ANN), Long Short-Term Memory (LSTM), and XGBoost models. These results indicate the potential of supervised learning models for Raga identification in Carnatic music[26].

Raga Recognition in Indian Classical Music Using DeepLearning

In 2021, Shah, Devansh P., et al. Various studies have explored the application of deep learning techniques to recognize Ragas in Indian Classical Music. The fundamental idea behind this approach is to train a deep learning model by utilizing features extracted from different Ragas. Subsequently, this trained model is used to identify the Raga of a given audio signal. Multiple methodologies have been proposed for this task, including the use of raw spectrograms from the audio source and Long Short Term Memory-based Recurrent Neural Networks (LSTM-RNN) for learning temporal sequences in music data. One study achieved highly accurate Raga recognition results of 98.98% through a deep learning and signal processing-based approach on a subset of 10 Ragas in the CompMusic dataset. These findings highlight the potential of deep learning techniques for Raga recognition in Indian Classical Music[27]

Classification of Indian Classical Carnatic Music Based on Raga Using Deep Learning

In 2020 John, Siji, et al. In Several studies have investigated the potential of utilizing deep learning for Raga classification in Indian Classical Carnatic Music. The fundamental concept is to train a deep learning model with features extracted from different Ragas and then employ this trained model to classify the Raga of an audio signal. Different methodologies have been proposed for this task, including using raw spectrograms and using Long Short Term Memory based Recurrent Neural Networks (LSTM-RNN) for learning temporal sequences in music data. The accuracy of these models varies with the methodology employed and the number of Ragas considered. a testing accuracy of 98.98% was achieved using a deep learning and signal processing-based approach on a subset of 10 Ragas in the CompMusic dataset[28]

Chapter 3

3.0 Methodology

In audio processing, software such as Wavpad is often utilized to process audio files. For feature extraction, where the Mel-frequency cepstral coefficients (MFCC) are extracted from the audio files. These coefficients are then used as input for machine learning algorithms . The machine learning algorithms predict the raaga associated with an input audio file.

3.1: Data Set

Audio files of different Ragas are downloaded from youtube, Than all the unnecessary noise removed using wavpad soft and all the files are converted from MP3 to WAV

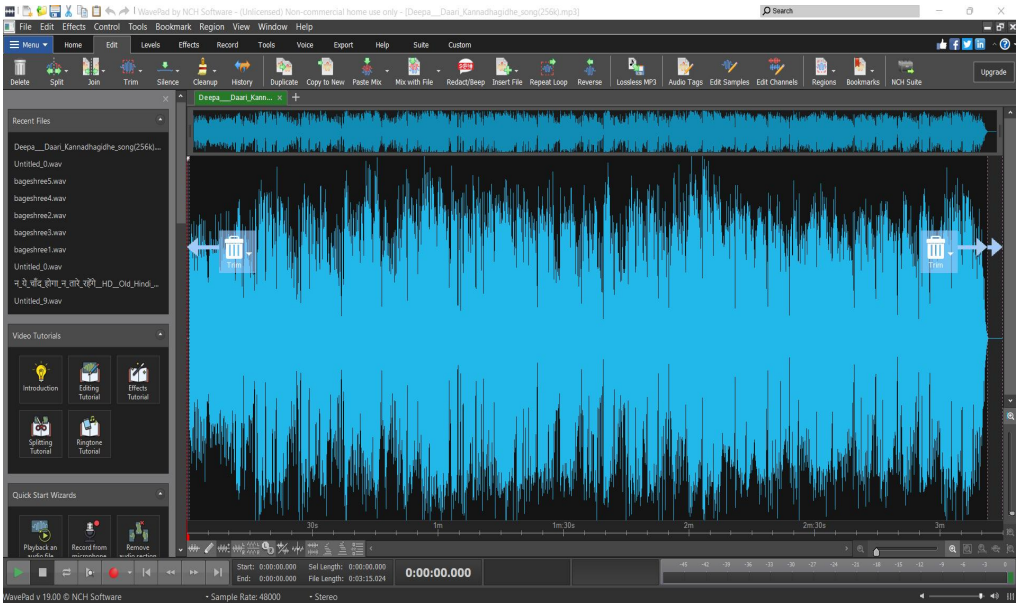

Figure 1: Audio pad software

Raaga Names	Total Samples	Vocals + Instruments	Vocals Without Instrument	Only Instruments
Bhairav	40	20	10	10
Marwa	40	20	10	10
Poorvi	40	20	10	10
Yaman	40	20	10	10
Bageshree	40	20	10	10

Table 4: Data segregation

5 Types Of Raagas Used In Dataset With Respect To Their Thaats

Sr.No	Thaat	Raaga	Sample Size	Sample length
1	Bhairav	Bhairav	40	3 Min
2	Marwa	Marwa	40	3 Min

3	Poorvi	Poorvi	40	3 Min
4	Kalyan	Yaman	40	3 Min
5	Kafi	Bageshree	40	3 Min

Table 5 :Type Of Ragas used in Dateset

Description Of Raagas

Raga Bhairav

Raag Bhairav is the very first Raag in Indian classical music; this is called Aadi raag. Bhairav is known as the king of early morning raags. It creates a meditative atmosphere that is majestic and masculine in character. If Bhairav were a divine person, he would be very serious and somber. This raag is a janak, or parent raag. It has produced many offspring raags, or prakaars (types) such as Ahir Bhairav, Aanand Bhairav, etc. Bhairav's equivalent in Carnatic music is Raag Mayamalavagowla. Aandolan (Oscillation) of komal Re and komal Dha swars; the timing of these movements is very important for rendering the personality of Bhairav[29].

Raag Marwa

Raag Marwa is the head of the Marwa family of Raags. Marwa traces its roots back to an ancient raag named Malawa. It sounds most beautiful at dusk. Marwa has the ability to

create a feeling of restlessness in both the performer as well as the audience by the over-use of komal Re and the deliberate avoidance of Sa. The restlessness is ultimately put to an end by displaying the Shadaj[30].

Raag Poorvi

Poorvi is a long-lived sunset raga from East India, which some describe as evoking a ‘serious mood of mystical contemplation’. Mixing narrow and wide intervals (all swaras have at least one immediate neighbour), its complex twists and turns belie the base scale’s neat, palindromic nature – with Sa and Pa sometimes being omitted or rendered durbal in ascent in order to ‘obscure’ these geometric balances, set amidst Ma-mixing phrase patterns such as rmG, GMPd, PMGmrG. Generally considered to have evolved from an archaic form of Bhairav (the same scale minus tivra Ma) , the raga’s modern incarnation is proximate to Puriya Dhanashree (which omits shuddha ma) – and prakriti with Paraj, Prabhat Bhairav, and Lalit Pancham‘s komal dha incarnation. Also see overlapping ragas including Reva, Purba, and Baradi[31].

Raga Yaman

Yaman is a Janak Raag from the Kalyan family of raags. Some texts use Raag Yaman and Raag Kalyan interchangeably. There is also a version which uses both shuddh and teevra Ma. It is called Yaman-Kalyan or Jaimini-Kalyan. Kalyan has been mentioned in Bharat Muni’s Natyashastra, which is an ancient literature on the performing arts, written 2000 years ago. It is called Kalyani in Carnatic music and is similar to the western Lydian scale[32]

Raga Bageshri

Raga Bageshri is a Hindustani classical raga also called Bagesari, Bageshwari, Vageshwari amongst others. It is hundreds of years old and an ocean of a raga, a combination of the ragas Dhanashri and Kanada, and a standard of music schools. Bageshri has a huge depth of expression allowing a never ending sea of exploration. It is mentioned in the older treatises and is said to have been first sung by Miyan Tansen, in the sixteenth century. A late night raag of the second prahar 9pm-12am, which is meant to depict the emotion of a woman waiting for reunion with her lover.With a very north Indian in flavour, the predominant mood shringar of Bageshri is teasing, romantic and flirting delight, It can also be used to express a variety of emotions, yearning, lost love and a touch of sadness.. A highly romantic raga, lending itself to often sad-romantic beauty, and the depths of unfulfilled love[33]**.**

3.2: BLOCK DIAGRAM

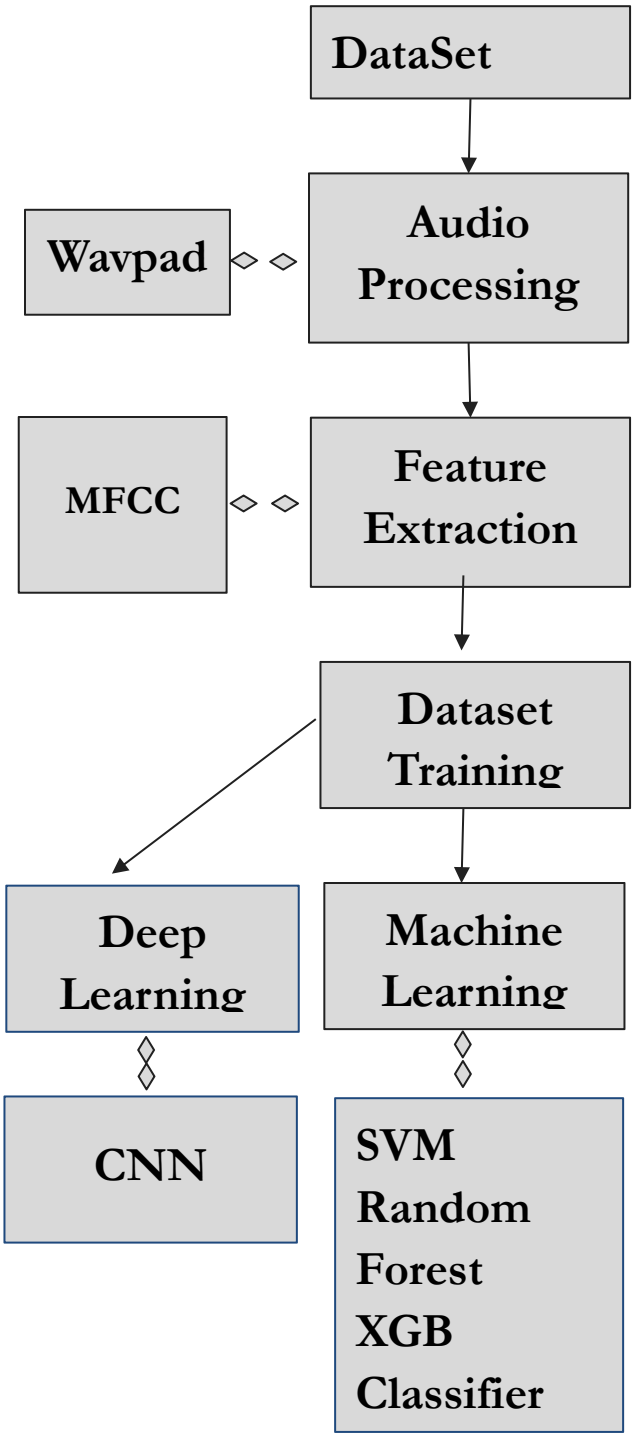

Figure 2 Block diagram

Neural networks

Neural networks are a commonly used, specific class of machine learning algorithms. Artificial neural networks are modeled on the human brain, in which thousands or millions of processing nodes are interconnected and organized into layers.

In an artificial neural network, cells, or nodes, are connected, with each cell processing inputs and producing an output that is sent to other neurons. Labeled data moves through the nodes, or cells, with each cell performing a different function. In a neural network trained to identify whether a picture contains a cat or not, the different nodes would assess the information and arrive at an output that indicates whether a picture features a cat.

Deep learning

Deep learning networks are neural networks with many layers. The layered network can process extensive amounts of data and determine the "weight" of each link in the network. For example, in an image recognition system, some layers of the neural network might detect individual features of a face, like eyes, nose, or mouth. In contrast, another layer would be able to tell whether those features appear in a way that indicates a face. Like neural networks, deep learning is modeled on the way the human brain works and powers many machine learning uses, like autonomous vehicles, chatbots, and medical diagnostics[4].

"The more layers you have, the more potential you have for doing complex things well, " Malone said.

Deep learning requires a great deal of computing power, which raises concerns about its economic and environmental sustainability.

CNN (Convolutional Neural Networks)

Deep Learning (DL) is the latest achievement of the Machine Learning era where it has presented near-human initially, and nowadays super-human abilities in many applications including voice-to-text translations, object detection and recognition, anomaly detection, recognizing emotions from audio or video recordings, etc. Even before the introduction of the AlexNet, perhaps one can consider that this era has begun with the ground-breaking article published in the journal, *Science*, in 2006 by Hinton and Salakhutdinov [59], which explained the role of "the depth" of an ANN in machine learning. It basically points out the fact that ANNs with several hidden layers can have a powerful learning ability, which can further be improved with the increasing depth –or equivalently the number of hidden layers. Hence comes the term "Deep" learning, a particular ML branch, which can tackle complex patterns and objects in massive size datasets[35].

In this section, we shall begin with the fundamental tool of DL, the deep (and conventional) CNNs whilst explaining their basic features and blocks. We will briefly discuss the most popular deep CNNs ever proposed and then move on with the most recent CNN architecture, the 1D CNNs, which are focused solely on 1D signal and data repositories. The particular focus will be drawn on compact and adaptive 1D CNN models, which can promise certain advantages and superiorities over their deep 2D counterparts.

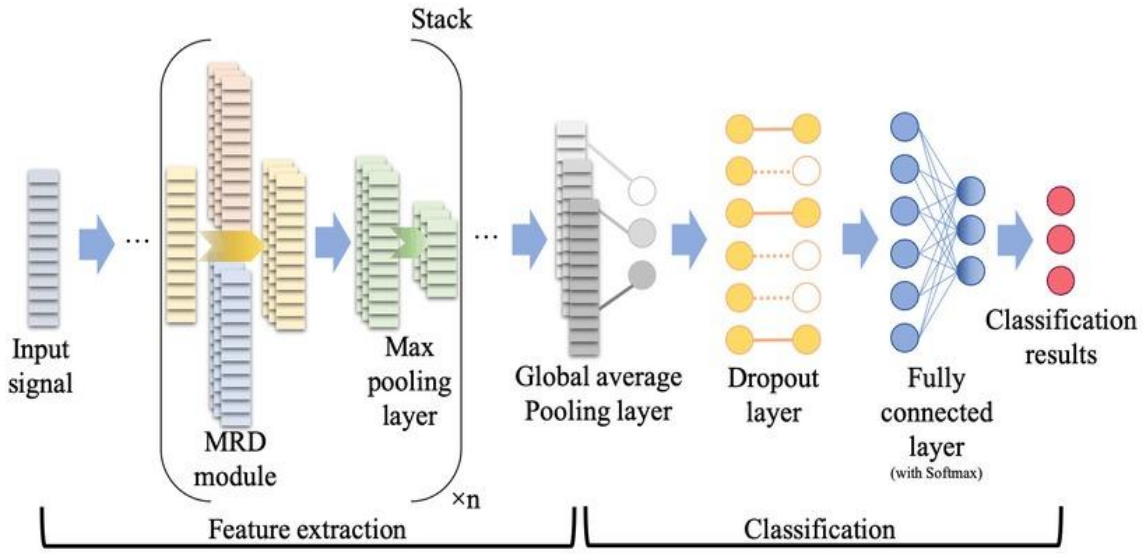

Figure 3 Convolutional Neural Networks

1D Convolutional Neural Networks

The conventional deep CNNs presented in the previous section are designed to operate exclusively on 2D data such as images and videos. This is why they are often referred to as, "2D CNNs". As an alternative, a modified version of 2D CNNs called 1D Convolutional Neural Networks (1D CNNs) have recently been developed . These studies have shown that for certain applications 1D CNNs are advantageous and thus preferable to their 2D counterparts in dealing with 1D signals due to the following reasons:

•There is a significant difference in terms of computational complexities of 1D and 2D convolutions, i.e., an image with *NxN* dimensions convolve with *KxK* kernel will have a computational complexity ~ O(*N2K2*) while in the corresponding 1D convolution (with the same dimensions, *N* and *K*) this is ~ O(*NK*).. This means that under equivalent

conditions (same configuration, network and hyper parameters) the computational complexity of a 1D CNN is significantly lower than the 2D CNN.

•As a general observation especially over the recent studies most of the 1D CNN applications have used compact (with 1–2 hidden CNN layers) configurations with networks having<10 K parameters whereas almost all 2D CNN applications have used "deep" architectures with more than 1 M (usually above 10 M) parameters. Obviously, networks with shallow architectures are much easier to train and implement.

•Usually, training deep 2D CNNs requires special hardware setup (e.g. Cloud computing or GPU farms). On the other hand, any CPU implementation over a standard computer is feasible and relatively fast for training compact 1D CNNs with few hidden layers (e.g. 2 or less) and neurons (e.g. < 50).

•Due to their low computational requirements, compact 1D CNNs are well-suited for real-time and low-cost applications especially on mobile or hand-held devices

In the aforementioned recent studies, compact 1D CNNs have demonstrated a superior performance on those applications which have a limited labeled data and high signal variations acquired from different sources (i.e., patient ECG, civil, mechanical or aerospace structures, high-power circuitry, power engines or motors, etc.). As illustrated in below Fig, two distinct layer types are proposed in 1D CNNs: 1) the so-called "CNN-layers" where both 1D convolutions, activation function and sub-sampling (pooling) occur, and 2) Fully-connected (dense) layers that are identical to the layers of a typical Multi-layer Perceptron (MLP) and therefore called as "MLP-layers". The configuration of a 1D-CNN is formed by the following hyper-parameters:

1)Number of hidden CNN and MLP layers/neurons (in the sample 1D CNN shown in below Fig, there are 3 and 2 hidden CNN and MLP layers, respectively).

2)Filter (kernel) size in each CNN layer (in the sample 1D CNN shown in below Fig, filter size is 41 in all hidden CNN layers).

3)Subsampling factor in each CNN layer (in the sample 1D CNN shown in below Fig , subsampling factor is 4).

4)The choice of pooling and activation functions.

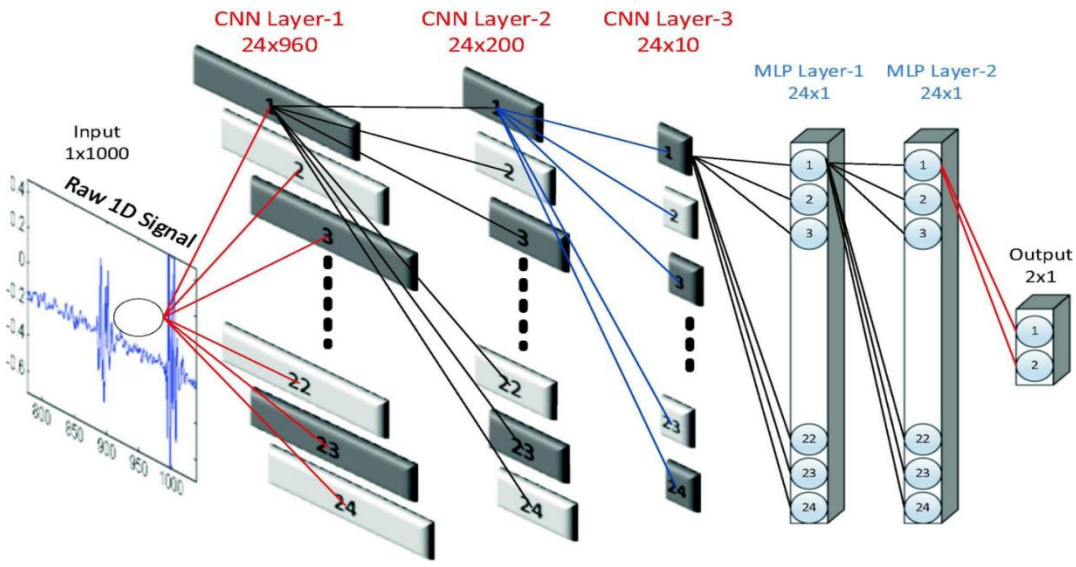

Figure 4 A sample 1D CNN configuration with 3 CNN and 2 MLP layers.

As in the conventional 2D CNNs, the input layer is a passive layer that receives the raw 1D signal and the output layer is a MLP layer with the number of neurons equal to the number of classes. Three consecutive CNN layers of a 1D CNN are presented in the below Fig . As shown in this figure, the 1D filter kernels have size 3 and the sub-sampling factor is 2 where the kth neuron in the hidden CNN layer, l, first performs a sequence of convolutions, the sum of which is passed through the activation function, ,

followed by the sub-sampling operation. This is indeed the main difference between 1D and 2D CNNs, where 1D arrays replace 2D matrices for both kernels and feature maps. As a next step, the CNN layers process the raw 1D data and "learn to extract" such features which are used in the classification task performed by the MLP-layers. As a consequence, both feature extraction and classification operations are *fused* into one process that can be optimized to maximize the classification performance. This is the major advantage of 1D CNNs which can also result in a low computational complexity since the only operation with a significant cost is a sequence of 1D convolutions which are simply linear weighted sums of two 1D arrays. Such a linear operation during the Forward and Back-Propagation operations can effectively be executed in parallel.

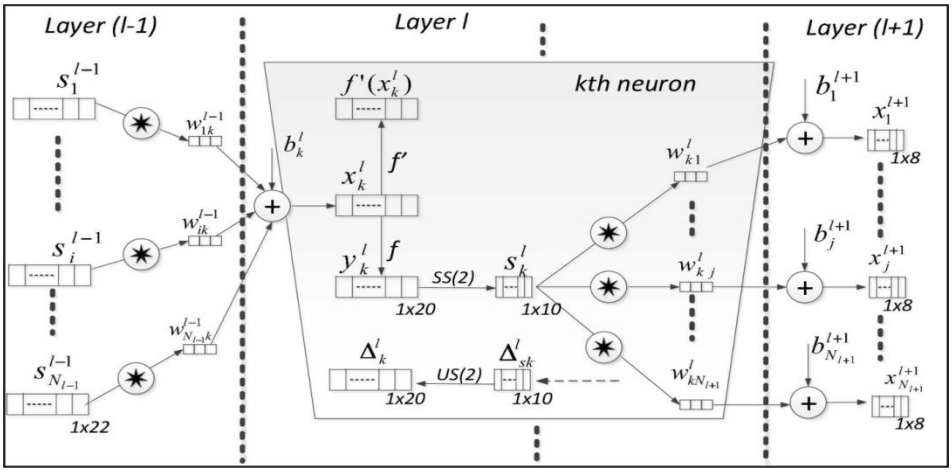

Figure 5 Three consecutive hidden CNN layers of a 1D CNN

This is also an adaptive implementation since the CNN topology will allow the variations in the input layer dimension in such a way that the sub-sampling factor of the output CNN layer is tuned adaptively. The details related to Forward and Back-Propagation in CNN layers are presented in the next sub-section.

3.3: Machine Learning

What is machine learning?

Machine learning is a subfield of artificial intelligence, which is broadly defined as the capability of a machine to imitate intelligent human behavior. Artificial intelligence systems are used to perform complex tasks in a way that is similar to how humans solve problems[36].

SVM(Support Vector Machine Algorithm)

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane[37].

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:

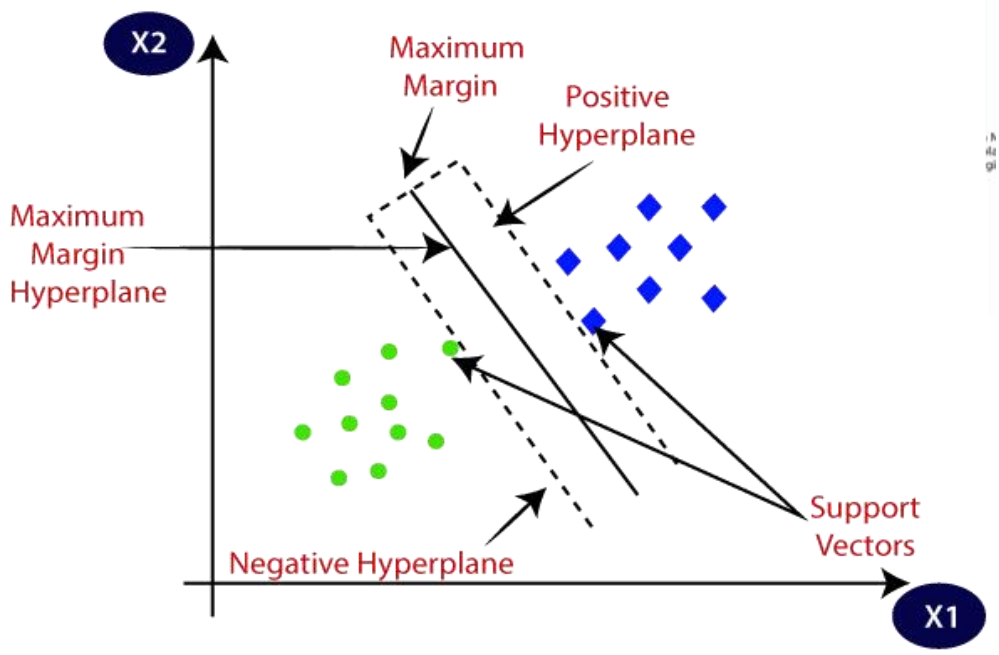

Figure 6 SVM Hyperplane

SVM can be of two types:

Linear SVM: Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

Non-linear SVM: Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

Hyperplane and Support Vectors in the SVM algorithm:

Hyperplane: There can be multiple lines/decision boundaries to segregate the classes in n dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.

The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2-dimension plane.

We always create a hyperplane that has a maximum margin, which means the maximum distance between the data points.

Support Vectors Regression:

The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

How does SVM works?

Linear SVM:

The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair(x1, x2) of coordinates in either green or blue. Consider the below image**:**

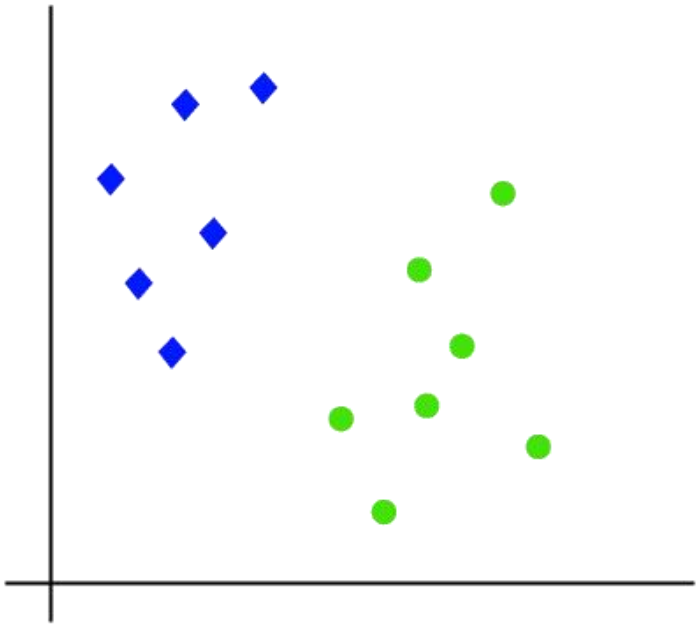

Figure 7Linearly separable data points

So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes. Consider the below

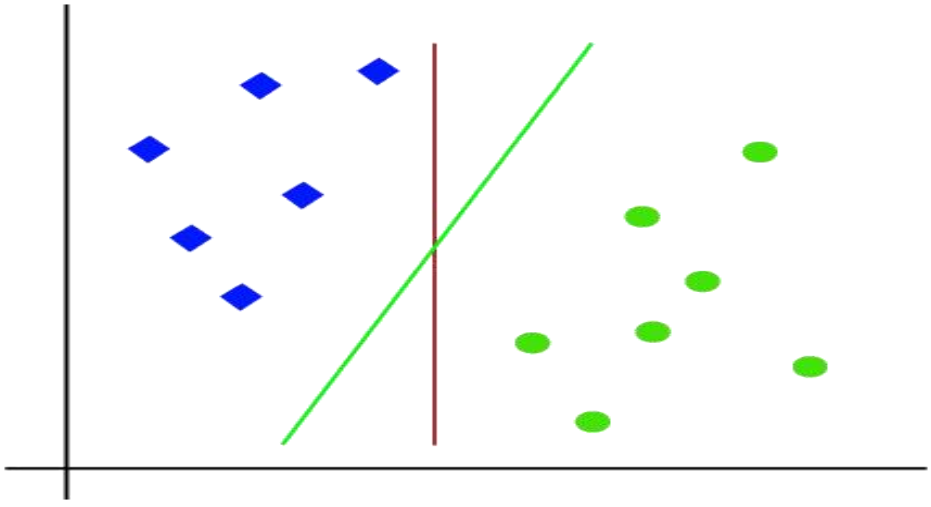

Figure 8 Linearly separable data points

Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a hyperplane. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane is called as margin. And the goal of SVM is to maximize this margin. The hyperplane with maximum margin is called the optimal hyperplane.

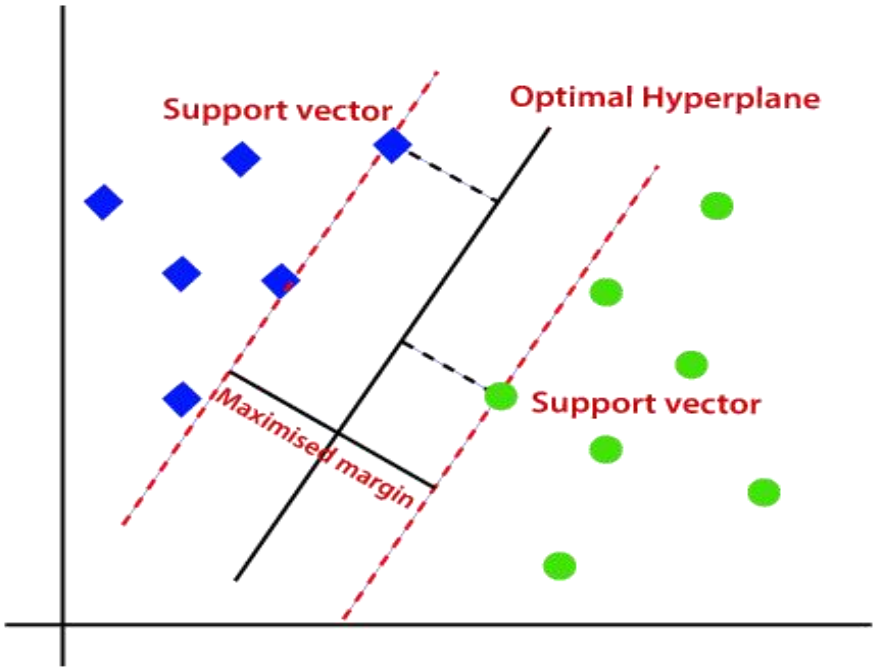

Figure 9:Optimal hyperplane

Non-Linear SVM:

If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line. Consider the below image:

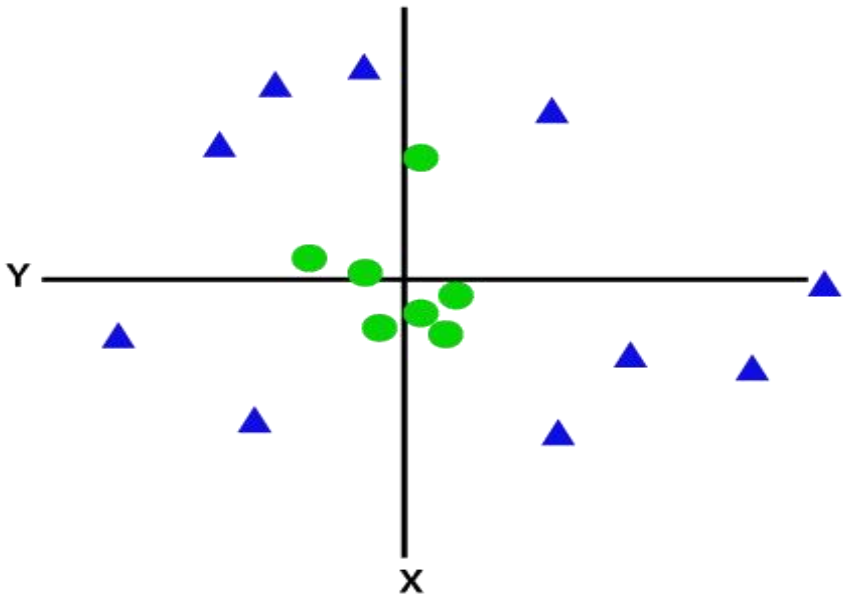

Figure 10 Non linear data

So to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third dimension z. It can be calculated as:

$$z=x^2+y^2$$

By adding the third dimension, the sample space will become as below image:

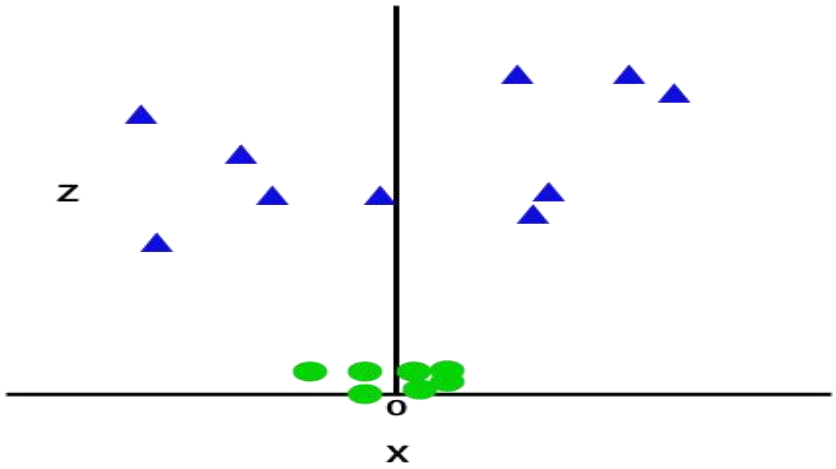

Figure 11 Mapping 1D data to 2D to become able to separate the two classes

So now, SVM will divide the datasets into classes in the following way. Consider the below image:

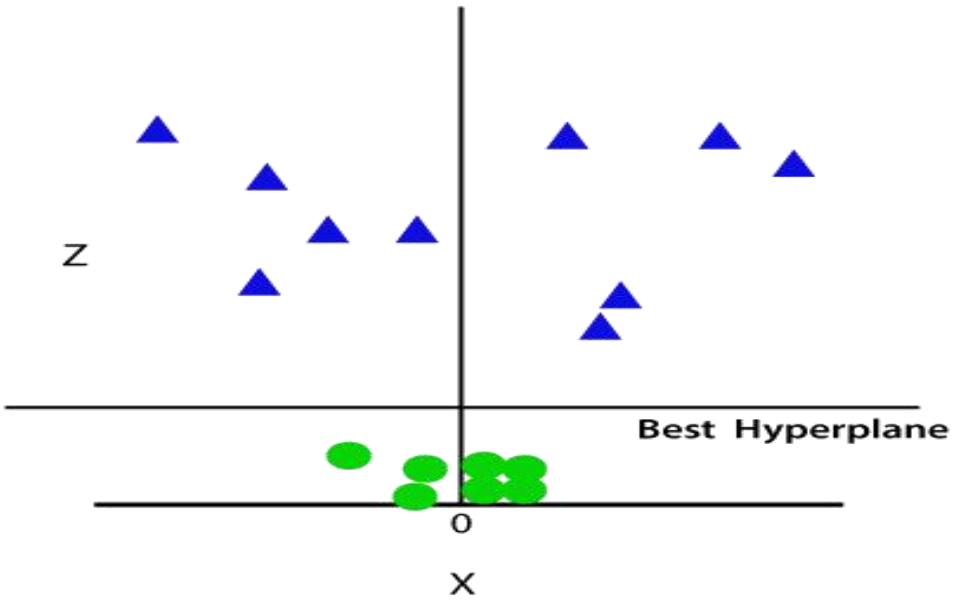

Figure 12 Best Hyperplane in 3D space

Since we are in 3-d Space, hence it is looking like a plane parallel to the x-axis. If we convert it in 2d space with z=1, then it will become as:

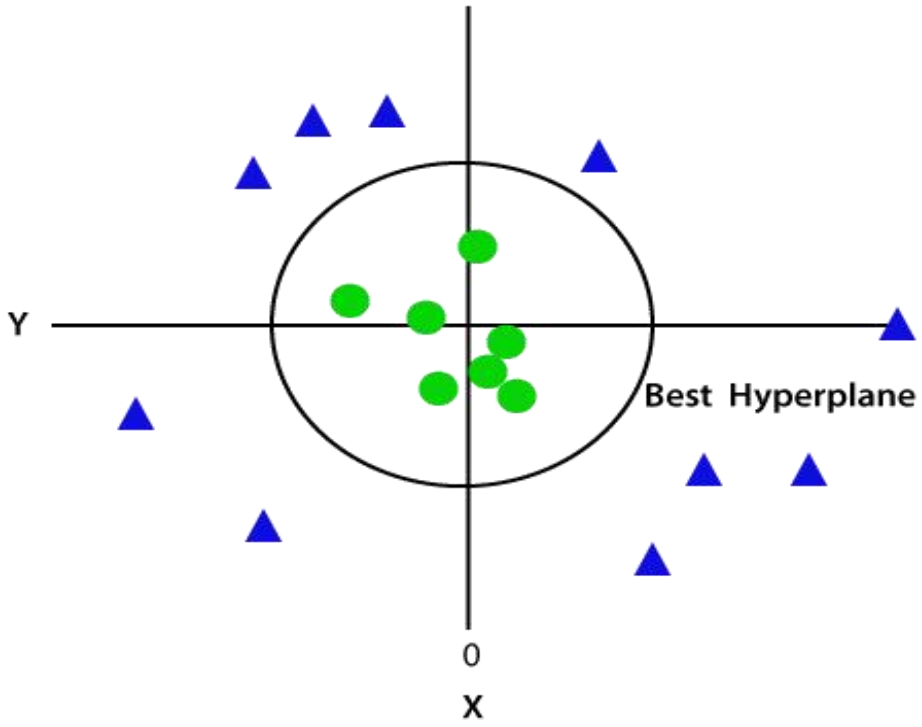

Figure 13 circumference of radius 1 in case of non-linear data

Hence we get a circumference of radius 1 in case of non-linear data.

Random Forest Algorithm

Random Forest algorithm is a powerful tree learning technique in Machine Learning. It works by creating a number of Decision Trees during the training phase. Each tree is constructed using a random subset of the data set to measure a random subset of features in each partition. This randomness introduces variability among individual trees, reducing the risk of overfitting and improving overall prediction performance. In prediction, the algorithm aggregates the results of all trees, either by voting (for classification tasks) or

by averaging (for regression tasks) This collaborative decision-making process, supported by multiple trees with their insights, provides an example stable and precise results. Random forests are widely used for classification and regression functions, which are known for their ability to handle complex data, reduce overfitting, and provide reliable forecasts in different environments[38].

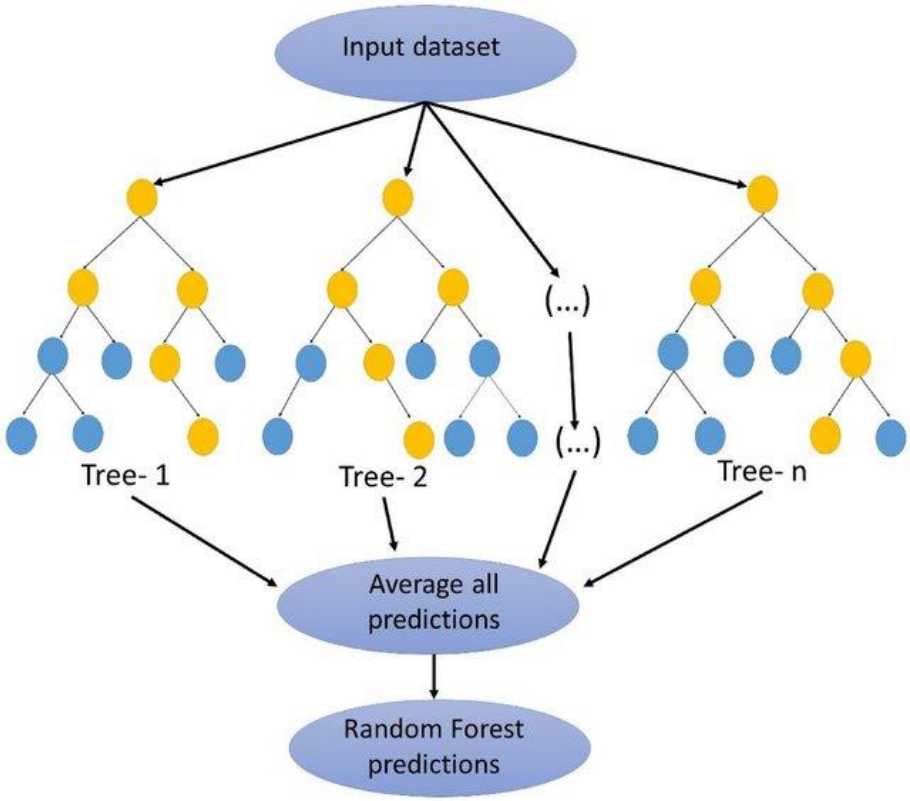

Figure 14 Random Forest Classifier

What are Ensemble Learning models?

Ensemble learning models work just like a group of diverse experts teaming up to make decisions – think of them as a bunch of friends with different strengths tackling a problem together. Picture it as a group of friends with different skills working on a

project. Each friend excels in a particular area, and by combining their strengths, they create a more robust solution than any individual could achieve alone.

Similarly, in ensemble learning, different models, often of the same type or different types, team up to enhance predictive performance. It's all about leveraging the collective wisdom of the group to overcome individual limitations and make more informed decisions in various machine learning tasks. Some popular ensemble models include- XGBoost, AdaBoost, LightGBM, Random Forest, Bagging, Voting etc.

What is Bagging and Boosting?

Bagging is an ensemble learning model, where multiple week models are trained on different subsets of the training data. Each subset is sampled with replacement and prediction is made by averaging the prediction of the week models for regression problem and considering majority vote for classification problem.

Boosting trains multiple based models sequentially. In this method, each model tries to correct the errors made by the previous models. Each model is trained on a modified version of the dataset, the instances that were misclassified by the previous models are given more weight. The final prediction is made by weighted voting.

How Does Random Forest Work?

The random Forest algorithm works in several steps which are discussed below

Ensemble of Decision Trees: Random Forest leverages the power of ensemble learning by constructing an army of Decision Trees. These trees are like individual experts, each specializing in a particular aspect of the data. Importantly, they operate independently, minimizing the risk of the model being overly influenced by the nuances of a single tree.

Random Feature Selection: To ensure that each decision tree in the ensemble brings a unique perspective, Random Forest employs random feature selection. During the training of each tree, a random subset of features is chosen. This randomness ensures that each tree focuses on different aspects of the data, fostering a diverse set of predictors within the ensemble.

Bootstrap Aggregating or Bagging: The technique of bagging is a cornerstone of Random Forest's training strategy which involves creating multiple bootstrap samples from the original dataset, allowing instances to be sampled with replacement. This results in different subsets of data for each decision tree, introducing variability in the training process and making the model more robust.

Decision Making and Voting: When it comes to making predictions, each decision tree in the Random Forest casts its vote. For classification tasks, the final prediction is determined by the mode (most frequent prediction) across all the trees. In regression tasks, the average of the individual tree predictions is taken. This internal voting mechanism ensures a balanced and collective decision-making process.

Xbg Classifier

XGBoost, short for eXtreme Gradient Boosting, is a powerful machine learning algorithm known for its efficiency, speed, and accuracy. It belongs to the family of boosting algorithms, which are ensemble learning techniques that combine the predictions of multiple weak learners. In this article, we will explore XGBoost step by step, building on existing knowledge with decision trees, boosting, and ensemble learning, XGBoost, or Extreme Gradient Boosting, is a state-of-the-art machine learning algorithm renowned for its exceptional predictive performance. It is the gold standard in ensemble

learning, especially when it comes to gradient-boosting algorithms. It develops a series of weak learners one after the other to produce a reliable and accurate predictive model.Fundamentally, XGBoost builds a strong predictive model by aggregating the predictions of several weak learners, usually decision trees. It uses a boosting technique to create an extremely accurate ensemble model by having each weak learner after it correct the mistakes of its predecessors[39].

The optimization method (gradient) minimizes a cost function by repeatedly changing the model's parameters in response to the gradients of the errors. The algorithm also presents the idea of "gradient boosting with decision trees," in which the objective function is reduced by calculating the importance of each decision tree that is added to the ensemble in turn. By adding a regularization term and utilizing a more advanced optimization algorithm, XGBoost goes one step further and improves accuracy and efficiency.

It has gained popularity and widespread usage because it can handle large datasets in a variety of machine-learning tasks, including regression and classification.

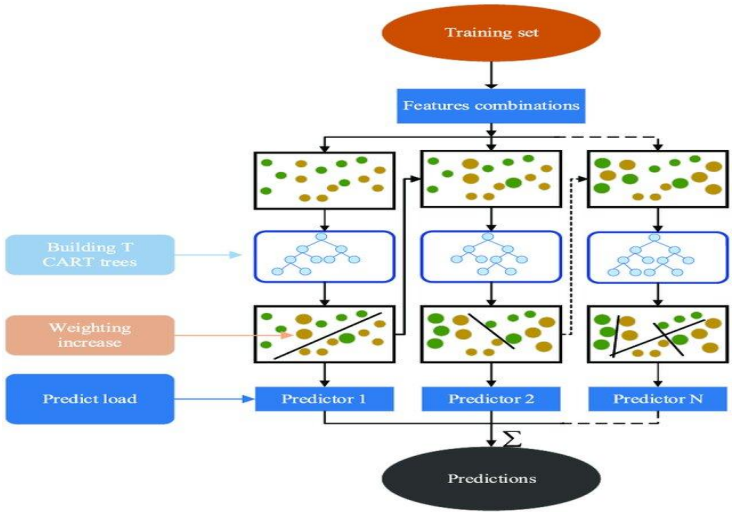

Figure 15 XGB Classifier

What Makes XGBoost “eXtreme”?

XGBoost extends traditional gradient boosting by including regularization elements in the objective function, XGBoost improves generalization and prevents overfitting.

Preventing Overfitting

The learning rate, also known as shrinkage, is a new parameter introduced by XGBoost. It is represented by the symbol “eta.” It quantifies each tree’s contribution to the total prediction. Because each tree has less of an influence, an optimization process with a lower learning rate is more resilient. By making the model more conservative, regularization terms combined with a low learning rate assist avoid overfitting.

XGBoost constructs trees level by level, assessing whether adding a new node (split) enhances the objective function as a whole at each level. The split is trimmed if not. This level growth along with trimming makes the trees easier to understand and easier to create.

The regularization terms, along with other techniques such as shrinkage and pruning, play a crucial role in preventing overfitting, improving generalization, and making XGBoost a robust and powerful algorithm for various machine learning tasks.

Tree Structure

Conventional decision trees are frequently developed by expanding each branch until a stopping condition is satisfied, or in a depth-first fashion. On the other hand, XGBoost builds trees level-wise or breadth-first. This implies that it adds nodes for every feature at a certain depth before moving on to the next level, so growing the tree one level at a time.

Determining the Best Splits: XGBoost assesses every split that might be made for every feature at every level and chooses the one that minimizes the objective function as much as feasible (e.g., minimizing the mean squared error for regression tasks or cross-entropy for classification tasks).

In contrast, a single feature is selected for a split at each level in depth-wise expansion.Prioritizing Important Features: The overhead involved in choosing the best split for each feature at each level is decreased by level-wise growth. XGBoost eliminates the need to revisit and assess the same feature more than once during tree construction because all features are taken into account at the same time. This is particularly beneficial when there are complex interactions among features, as the algorithm can adapt to the intricacies of the data.

Handling Missing Data

XGBoost functions well even with incomplete datasets because of its strong mechanism for handling missing data during training. To effectively handle missing values, XGBoost employs a “Sparsity Aware Split Finding” algorithm. The algorithm treats missing values as a separate value and assesses potential splits in accordance with them when determining the optimal split at each node. If a data point has a missing value for a particular feature during tree construction, it descends a different branch of the tree. The potential gain from splitting the data based on the available feature values—including missing values—is taken into account by the algorithm to determine the ideal split. It computes the gain for every possible split, treating the cases where values are missing as a separate group.

If the algorithm's path through the tree comes across a node that has missing values while generating predictions for a new instance during inference, it will proceed along the default branch made for instances that have missing values. This guarantees that the model can generate predictions in the event that there are missing values in the input data.

Cache-Aware Access in XGBoost

Cache memory located closer to the CPU offers faster access times, and modern computer architectures consist of hierarchical memory systems, By making effective use of this cache hierarchy, computational performance can be greatly enhanced. This is why XGBoost's cache-aware access was created, with the goal of reducing memory access times during the training stage. The most frequently accessed data is always available for computations because XGBoost processes data by storing portions of the dataset in the CPU's cache memory. This method makes use of the spatial locality principle, which states that adjacent memory locations are more likely to be accessed concurrently. Computations are sped up by XGBoost because it arranges data in a cache-friendly manner, reducing the need to fetch data from slower main memory.

Approximate Greedy Algorithm

This algorithm uses weighted quantiles to find the optimal node split quickly rather than analyzing each possible split point in detail. When working with large datasets, XGBoost makes the algorithm more scalable and faster by approximating the optimal split, which dramatically lowers the computational cost associated with evaluating all candidate splits.

Parameters in XGBoost

Learning Rate (eta): An important variable that modifies how much each tree contributes to the final prediction. While more trees are needed, smaller values frequently result in more accurate models.

Max Depth: This parameter controls the depth of every tree, avoiding overfitting and being essential to controlling the model's complexity.

Gamma: Based on the decrease in loss, it determines when a node in the tree will split. The algorithm becomes more conservative with a higher gamma value, avoiding splits that don't appreciably lower the loss. It aids in managing tree complexity.

Subsample: Manages the percentage of data that is sampled at random to grow each tree, hence lowering variance and enhancing generalization. Setting it too low, though, could result in underfitting.

Colsample Bytree: Establishes the percentage of features that will be sampled at random for growing each tree.

n_estimators: Specifies the number of boosting rounds.

lambda (L2 regularization term) and alpha (L1 regularization term): Control the strength of L2 and L1 regularization, respectively. A higher value results in stronger regularization.

min_child_weight: Influences the tree structure by controlling the minimum amount of data required to create a new node.

scale_pos_weight: Useful in imbalanced class scenarios to control the balance of positive and negative weights.

Why XGboost?

XGBoost is highly scalable and efficient as It is designed to handle large datasets with millions or even billions of instances and features. XGBoost implements parallel processing techniques and utilizes hardware optimization, such as GPU acceleration, to speed up the training process. This scalability and efficiency make XGBoost suitable for big data applications and real-time predictions.It provides a wide range of customizable parameters and regularization techniques, allowing users to fine-tune the model according to their specific needs.XGBoost offers built-in feature importance analysis, which helps identify the most influential features in the dataset. This information can be valuable for feature selection, dimensionality reduction, and gaining insights into the underlying data patterns. XGBoost has not only demonstrated exceptional performance but has also become a go-to tool for data scientists and machine learning practitioners across various languages. It has consistently outperformed other algorithms in Kaggle competitions, showcasing its effectiveness in producing high-quality predictive models.

MFCC (Mel Frequency Cepstral Coefficients) INTERNAL BLOCK

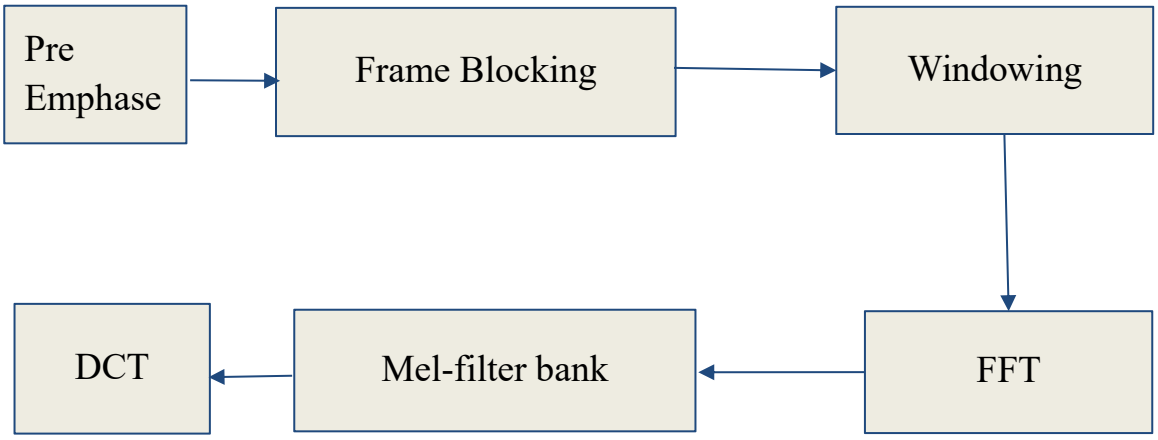

Figure 16 MFCC Internal Block Diagram

Mel Frequency Cepstral Coefficients (MFCCs) are a widely used feature extraction technique in audio and speech processing

What are MFCCs?

MFCCs are coefficients that collectively make up a Mel Frequency Cepstrum (MFC), which is a representation of the short-term power spectrum of a sound. They are used to represent the spectral characteristics of sound in a way that is well-suited for various machine learning tasks, such as speech recognition and music analysis[40].

Why are MFCCs important? The sounds generated by a human are filtered by the shape of the vocal tract including tongue, teeth etc. This shape determines what sound comes out. If we can determine the shape accurately, this should give us an accurate representation of the phoneme being produced1.

How are MFCCs calculated?

The MFCC feature extraction process is basically a 6-step process3:

1. Frame the signal into short frames: Frequencies in a signal change over time, so it doesn't make sense to do the Fourier transform across the entire signal. We frame the signal into 20–40 ms frames. 25ms is standard.

2. Windowing: Windowing is applied to notably counteract the assumption made by the Fast Fourier Transform that the data is infinite and to reduce spectral leakage3.

3. Calculation of the Discrete Fourier Transform: An NN-point FFT is done on each frame to calculate the frequency spectrum, which is also called Short-Time Fourier-Transform (STFT), where NN is typically 256 or 5123.

4. Applying Filter Banks: This step is to mimic the logarithmic perception of loudness and pitch of human auditory system and to decorrelate the energy values3.

5. Take the logarithm of all filterbank energies3.

6. Take the DCT of the log filterbank energies: Keep DCT coefficients 2-13, discard the rest3.

What do MFCCs represent?

If a cepstral coefficient has a positive value, the majority of the spectral energy is concentrated in the low-frequency regions. On the other hand, if a cepstral coefficient has a negative value, it represents that most of the spectral energy is concentrated at high frequencies.

Chapter 4

4. Analysis And Conclusions

4.1 Results

Table 5 shows the accuracy that the methods obtained for different Machine Learning models as well as Deep Learning models. It shows results of percentage accuracy obtained for raga identification for 5 Raagas each having 40 samples for respective (3min) duration in algorithmic approaches. The accuracy of 94.99% for CNN . Similarly 90%,80.5%,91% accuracy for SVM, XGB classifier, Random Forest result obtained . The initial experiments revealed that methods resulted in 100% (5 out of 5) detection of the given raga with the clip of 3 min duration for the samples chosen of different ragas.

Table Of Different Model Classifiers

Models	Training /Testing	Accuracy	F1-score	Precision	Recall	Cross-Validation
SVM	80/20	90	90	88	88	88
	70/30	88	83	84	83	85
	60/40	77	78	79	78	82

CNN	80/20	94.99	94	88	88	70
	70/30	93.33	93.3	93.5	93.3	85
	60/40	89	89	90.7	90	80
XGB	80/20	75	92.5	93.36	92.5	88.12
	70/30	80.5	92.5	93.36	92.5	85
	60/40	76.6	92.5	93.36	92.5	82.5
RANDOM FOREST	80/20	90	88	88	88	88
	70/30	91	84	91	90	90
	60/40	83	83	84	82	85

Table 6 Table Of Different Model Classifiers

Confusion Matrix

Confusion matrix presents a table layout of the different outcomes of the prediction and results of a classification problem and helps visualize its outcomes. It plots a table of all the predicted and actual values of a classifier Calculations using Confusion Matrix:

We can perform various calculations for the model, such as the model's accuracy, using this matrix. These calculations are given below:

Classification Accuracy: It is one of the important parameters to determine the accuracy of the classification problems. It defines how often the model predicts the correct output.

It can be calculated as the ratio of the number of correct predictions made by the classifier to all number of predictions made by the classifiers. The formula is given below:

$$\text{Accuracy} = \frac{\text{TP} + \text{FP}}{\text{TP} + \text{FN} + \text{TN} + \text{FP}}$$

Misclassification rate: It is also termed as Error rate, and it defines how often the model gives the wrong predictions. The value of error rate can be calculated as the number of incorrect predictions to all number of the predictions made by the classifier. The formula is given below:

$$\text{ErrorRate} = \frac{\text{FP} + \text{FN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

Precision: It can be defined as the number of correct outputs provided by the model or out of all positive classes that have predicted correctly by the model, how many of them were actually true. It can be calculated using the below formula:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Recall: It is defined as the out of total positive classes, how our model predicted correctly. The recall must be as high as possible.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

F-measure: If two models have low precision and high recall or vice versa, it is difficult to compare these models. So, for this purpose, we can use F-score. This score helps us to evaluate the recall and precision at the same time. The F-score is maximum if the recall is equal to the precision. It can be calculated using the beType equation here.low formula:

$$\mathrm{F-measure} = \frac{2 * \mathrm{Recall} * \mathrm{Precision}}{\mathrm{Recall} + \mathrm{Precision}}$$

Result of Confusion Matrix

CONFUSION MATRIX FOR CNN

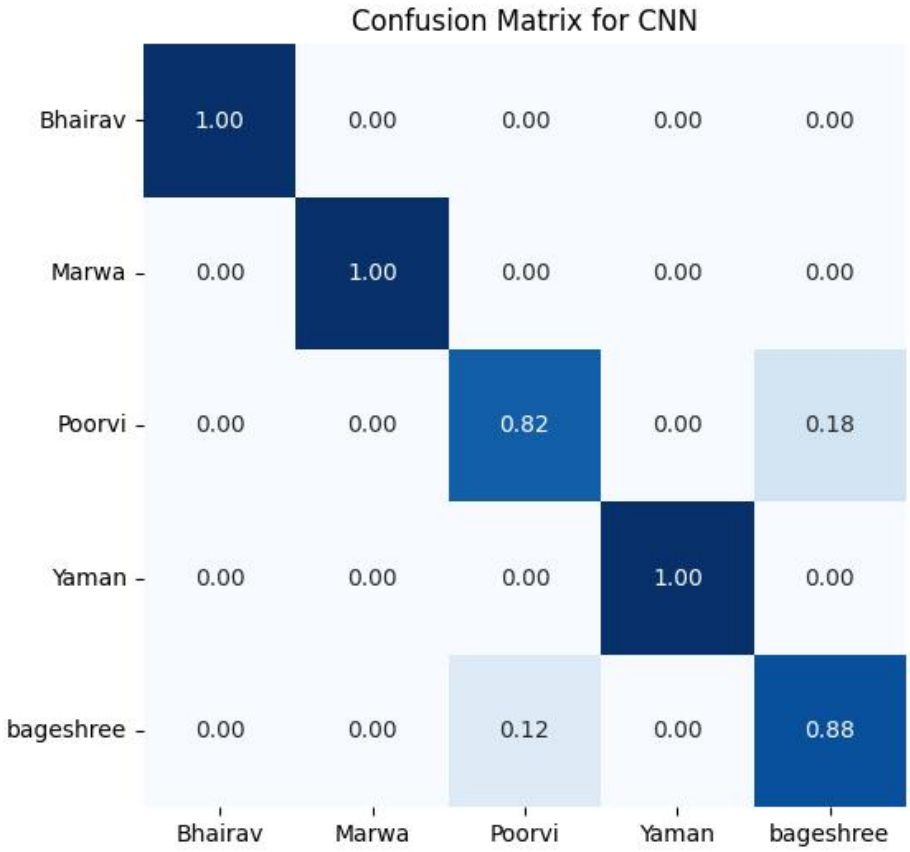

Figure 17 Confusion Matrix For CNN

The confusion matrix is derived from the CNN method, utilizing a dataset comprising 200 samples distributed among 5 classes.. 80% of the samples (160) were used for training, while the remaining 20% (40 samples) were reserved for testing. This means that each class is represented by 8 samples during the testing phase. The classification of 'Bhairav' , 'Marwa' and 'Yaman' achieved a perfect 100% prediction accuracy, whereas 'Poorvi' achieved 82% accuracy, misidentifying 18% with 'Bageshree'. 'Bageshree' achieved 88% accuracy, misidentifying 12% with 'Poorvi'.

CONFUSION MATRIX FOR SVM

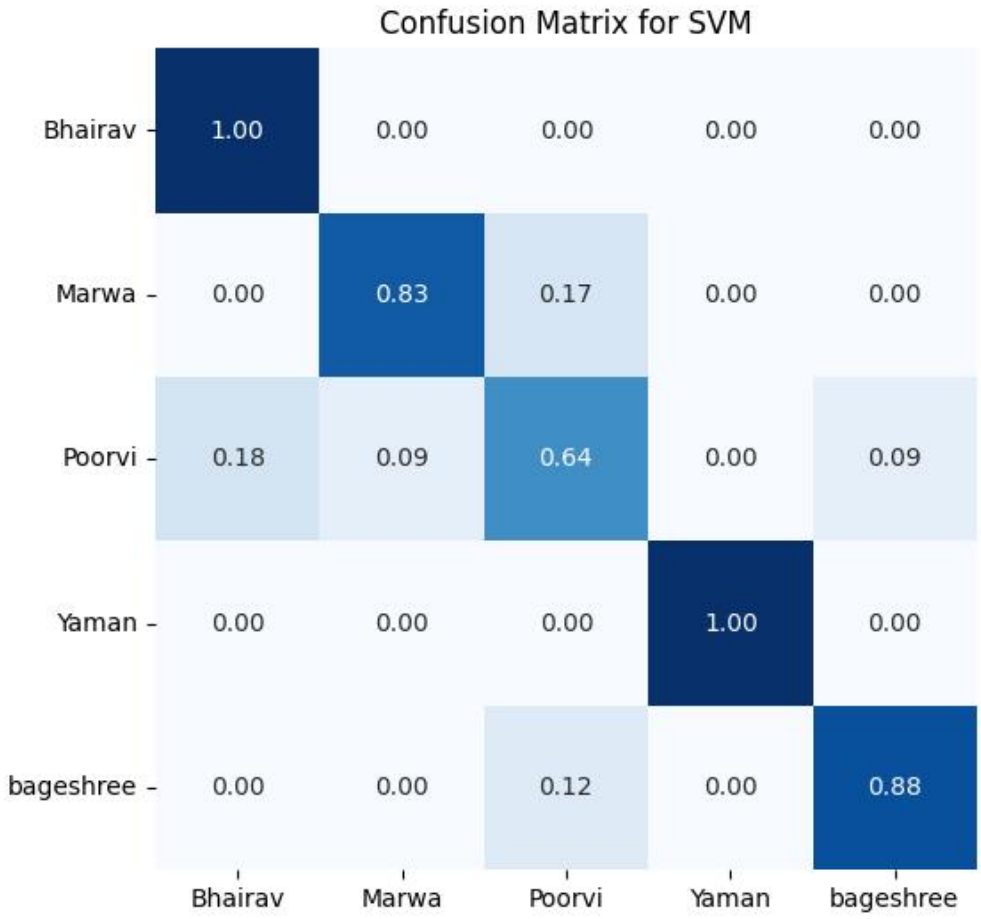

Figure 18 Confusion Matrix For SVM

The confusion matrix is derived from the SVM method, utilizing a dataset comprising 200 samples distributed among 5 classes.. 80% of the samples (160) were used for training, while the remaining 20% (40 samples) were reserved for testing. This means that each class is represented by 8 samples during the testing phase. The classification of 'Bhairav' and 'Yaman' achieved a perfect 100% prediction accuracy, whereas ’Marwa’ achived 83%,misidentifying 17% with 'Poorvi'.'Poorvi' achieved 64% accuracy, misidentifying 18% with 'Bhairav' and 9% with ‘Marwa’. 'Bageshree' achieved 88% accuracy, misidentifying 12% with 'Poorvi'.

CONFUSION MATRIX FOR XGB CLASSIFIER

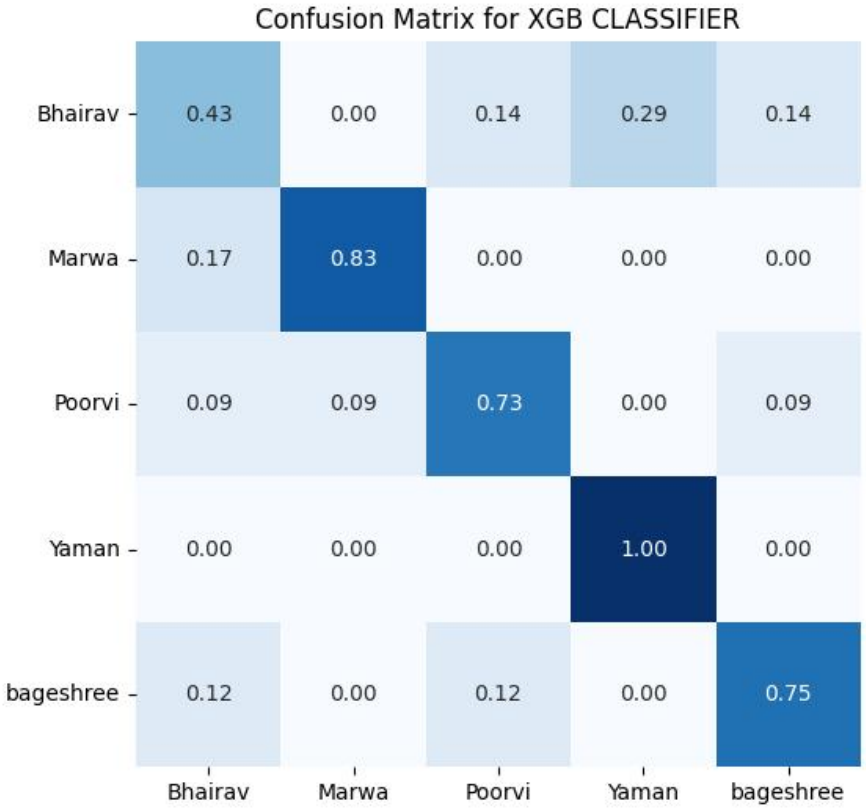

Figure 19 Confusion Matrix For XGB Classifier

The confusion matrix is derived from the XGB Classifier method, utilizing a dataset comprising 200 samples distributed among 5 classes.. 80% of the samples (160) were used for training, while the remaining 20% (40 samples) were reserved for testing. This means that each class is represented by 8 samples during the testing phase. The classification of 'Yaman' achieved a perfect 100% prediction accuracy, whereas 'Bhairav' achieved 43% accuracy, misidentifying 14% with 'Bageshree'.,29% with ‘Yaman’ and 14% with ’Poorvi’.’Marwa’ achieved 83%, misidentifying 17% . ‘Poorvi’ achieved 73% misidentifying 9% each with 'Bhairav','Marwa’ and 'Bageshree' . 'Bageshree' achieved 75% accuracy, misidentifying 12% with 'Poorvi' and 12% with ‘Bhairav’.

CONFUSION MATRIX FOR RANDOM FOREST

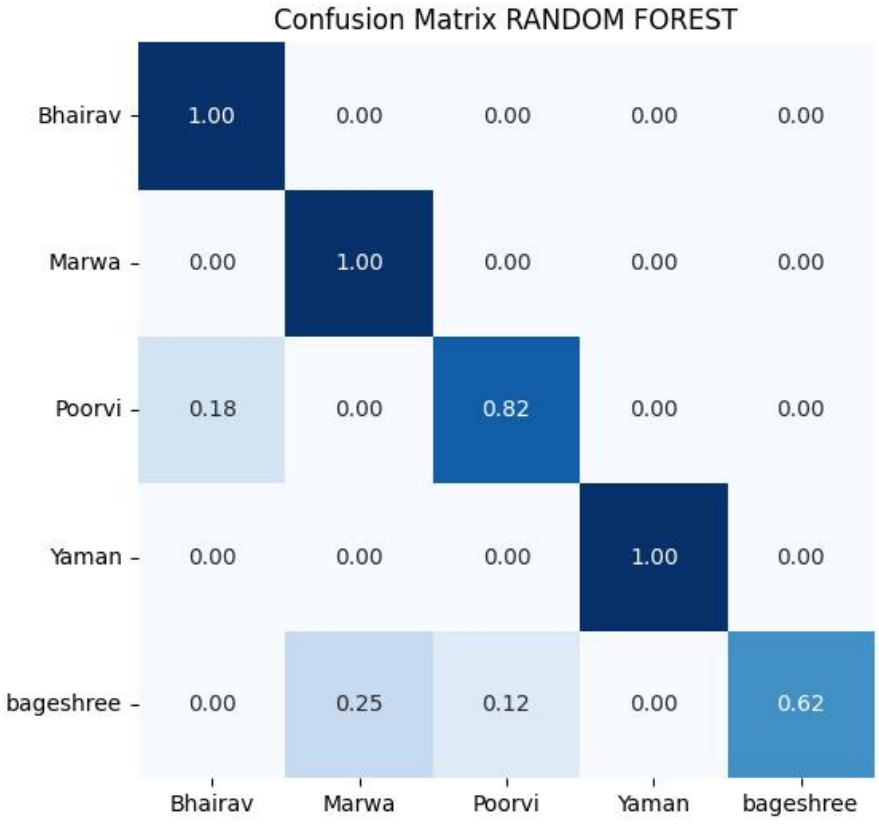

Figure 20 Confusion Matrix For Random Forest

The confusion matrix is derived from the Random Forest method, utilizing a dataset comprising 200 samples distributed among 5 classes.. 80% of the samples (160) were used for training, while the remaining 20% (40 samples) were reserved for testing. This means that each class is represented by 8 samples during the testing phase. The classification of 'Bhairav' , 'Marwa' and 'Yaman' achieved a perfect 100% prediction accuracy, whereas 'Poorvi' achieved 82% accuracy, misidentifying 18% with 'Bageshree'. 'Bageshree' achieved 62% accuracy, misidentifying 12% with 'Poorvi' and 25% with ‘Marwa’.

4.2 Conclusion

This work integrated a dataset containing 5 different RAAGAS Bandish audio samples to identify different RAAGAs in Indian Classical Music . This study concluded that the application of Deep learning gave better results for identifying RAAGAS than machine learning. The performance is evaluated with respect to Accuracy and F1-score. MFCC features give results based on timbre information of the signal. The ICM Raga identification is mainly performed based on patterns of Swaras and their variations. The patterns and variations of Swaras of ICM are well described by MFCC features. MFCC gives spectral information.The obtained accuracy is 94.99% for CNN . Similarly 90%, 80.5%, 91% accuracy for SVM, XGB classifier and Random Forest

4.3 Future Work

Future scope of this work can be aimed to Identify more different Ragas as well as Ragas from same Thaats and also to identify multiples Ragas. We can also work more feature extraction methods and can classify the best feature extraction method , for Data set we can try with different sample length like 5 min for each audio sample (Raga).To work on hardware to design a system which can record the Ragas and also predict it. Also futher we can work on music recommendation system based on raaga

4.4 REFERENCE

[1]https://icmacy.org/bageshri/#:~:text=Raag%20Bageshri%2DDebabrata%20Pal,and%20a%20touch%20of%20sadness

[2] https://saayujya.com/index.php/2021/03/23/thaats-in-hindustani-music-system/

[3] https://www.indianclassicalmusic.com/what-is-raag

[4]https://www.ibm.com/topics/deep-learning#:~:text=Deep%20learning%20is%20a%20subset,AI)%20in%20our%20lives%20today.

[5]Velankar, M., Deshpande, A., & Kulkarni, P. (2021). Melodic pattern recognition in Indian classical music for raga identification. International Journal of Information Technology, 13(1), 251–258. https://doi.org/10.1007/s41870-018-0245-6

[6]Waghmare, K. C., & Sonkamble, B. A. (2019). Analyzing acoustics of indian music audio signal using timbre and pitch features for raga identification. 2019 3rd International Conference on Imaging, Signal Processing and Communication (ICISPC), 42–46. https://doi.org/10.1109/ICISPC.2019.8935707

[7]Shirude, S. B., & Kolhe, S. R. (2024). Recognizing raga of indian classical songs using regular expressions. In A. J. Kulkarni & N. Cheikhrouhou (Eds.), Intelligent Systems for Smart Cities (pp. 367–383). Springer Nature Singapore. https://doi.org/10.1007/978-981-99-6984-5_24

[8]Pandey, Gaurav & Mishra, Chaitanya & Ipe, Paul. (2003). TANSEN: A System for Automatic Raga Identification.. 1350-1363.

[9]Anand, A. (2019). Raga identification using convolutional neural network. 2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP), 1–6. https://doi.org/10.1109/ICACCP.2019.8882942

[10]Deshmukh, Kavita. M., & Deore, P. J. (2017). Raga Identification Using MFCC and Chroma features. 8(5), 725–729. https://doi.org/10.26483/ijarcs.v8i5.340

[11]Kiran, & S, S. K. D. (2022). Machine learning based indian raga identification for music therapy. https://doi.org/10.21203/rs.3.rs-1760645/v2

[12]Pradeep, R., Dhara, P., Rao, K. S., & Dasgupta, P. (2015). Raga identification based on Normalized Note Histogram features. 2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 1491–1496. https://doi.org/10.1109/ICACCI.2015.7275823

[13]Lele, J. A., & Abhyankar, A. S. (2019). Towards raga identification of hindustani classical music. 2019 IEEE Pune Section International Conference (PuneCon), 1–4. https://doi.org/10.1109/PuneCon46936.2019.9105894

[14]Gowrishankar, B. S., & U. Bhajantri, N. (2023). Raga classification using enhanced spatial bound whale optimization algorithm. Indonesian Journal of Electrical Engineering and Computer Science, 30(2), 825. https://doi.org/10.11591/ijeecs.v30.i2.pp825-837

[15]Joshi, D., Pareek, J., & Ambatkar, P. (2021). Indian Classical Raga Identification using Machine Learning. International Symposium on Intelligent Control.

[16]Chhetri, A. R., Kumar, K., Muthyala, M. P., M R, S., & Bangalore, R. A. (2023). Carnatic music identification of melakarta ragas through machine and deep learning using audio signal processing. 2023 4th International Conference for Emerging Technology (INCET), 1–5. https://doi.org/10.1109/INCET57972.2023.10170568

[17]Gowrishankar, Bettadamadahally Shivakumaraswamy, and Nagappa U. Bhajantri. 'Raga Classification Using Enhanced Spatial Bound Whale Optimization Algorithm'. Indonesian Journal of Electrical Engineering and Computer Science, vol. 30, no. 2, May 2023, p. 825. DOI.org (Crossref), https://doi.org/10.11591/ijeecs.v30.i2.pp825-837.

[18]Singha, Anupam, et al. 'Deep Learning-Based Classification of Indian Classical Music Based on Raga'. 2023 6th International Conference on Information Systems and Computer Networks (ISCON), IEEE, 2023, pp. 1–7. DOI.org (Crossref), https://doi.org/10.1109/ISCON57294.2023.10111985.

[19]Sharma, Ashish, and Ambuja Salgaonkar. 'Raga Recognition Using Neural Networks and N-Grams of Melodies'. Computer Assisted Music and Dramatics, edited by Ambuja Salgaonkar and Makarand Velankar, vol. 1444, Springer Nature Singapore, 2023, pp. 93–109. DOI.org (Crossref), https://doi.org/10.1007/978-981-99-0887-5_7.

[20]K, Anitha, and Parameshachari B. D. Machine Learning Based Indian Raga Identification for Music Therapy. 30 June 2022. In Review, https://doi.org/10.21203/rs.3.rs-1760645/v1.

[21]Kavitha, P., et al. ‘A Deep Learning Based Approach for Janya Raga Classification in Carnatic Music’. 2022 6th SLAAI International Conference on Artificial Intelligence (SLAAI-ICAI), IEEE, 2022, pp. 1–6. DOI.org (Crossref), https://doi.org/10.1109/SLAAI-ICAI56923.2022.10002700.

[22]Hebbar, Devayani, and Vandana Jagtap. A Comparison of Audio Preprocessing Techniques and Deep Learning Algorithms for Raga Recognition. 2022. DOI.org (Datacite), https://doi.org/10.48550/ARXIV.2212.05335.

[23]Rajan, Rajeev, and Sreejith Sivan. ‘Raga Recognition in Indian Carnatic Music Using Convolutional Neural Networks’. WSEAS TRANSACTIONS ON ACOUSTICS AND MUSIC, vol. 9, May 2022, pp. 5–10. DOI.org (Crossref), https://doi.org/10.37394/232019.2022.9.2.

[24]P B, Manjunatha, et al. ‘RAGA DETECTION’. International Research Journal of Computer Science, vol. 9, no. 8, Aug. 2022, pp. 245–49. DOI.org (Crossref), https://doi.org/10.26562/irjcs.2022.v0908.18.

[25]Department of E&TC, Modern College of Engineering, Pune (Maharashtra), India., et al. ‘Raga Classification Based on Novel Method of Pitch Co-Occurrence’. International Journal of Recent Technology and Engineering (IJRTE), vol. 11, no. 1, May2022,pp.23–27.DOI.org(Crossref), https://doi.org/10.35940/ijrte.A6886.0511122.

[26]Patil, Surekha, et al. ‘A Survey on Computational Approaches for Raga Identification’. 2022 IEEE World Conference on Applied Intelligence and Computing

(AIC), IEEE, 2022, pp. 85–92. DOI.org (Crossref), https://doi.org/10.1109/AIC55036.2022.9848984.

[27]Kaimal, Veena, and Snehlata Barde. 'Implementation of a Supervised Learning Model for Raga Identification in Carnatic Music'. 2021 Asian Conference on Innovation in Technology (ASIANCON), IEEE, 2021, pp. 1–6. DOI.org (Crossref), https://doi.org/10.1109/ASIANCON51346.2021.9544628.

[28]Shah, Devansh P., et al. 'Raga Recognition in Indian Classical Music Using Deep Learning'. Artificial Intelligence in Music, Sound, Art and Design, edited by Juan Romero et al., vol. 12693, Springer International Publishing, 2021, pp. 248–63. DOI.org (Crossref), https://doi.org/10.1007/978-3-030-72914-1_17.

[29]John, Siji, et al. 'Classification of Indian Classical Carnatic Music Based on Raga Using Deep Learning'. 2020 IEEE Recent Advances in Intelligent Computational Systems (RAICS), IEEE, 2020, pp. 110–13. DOI.org (Crossref), https://doi.org/10.1109/RAICS51191.2020.9332482.

[30] https://www.saregama.com/song/raga-yaman_129268

[31] https://www.saregama.com/song/raag-ahir-bhairav_128915

[32] https://www.indianclassicalmusic.com/marwa

[33] https://ragajunglism.org/ragas/poorvi/

[34]https://icmacy.org/bageshri/#:~:text=Raag%20Bageshri%2DDebabrata%20Pal,and%20a%20touch%20of%20sadness

[35] https://www.sciencedirect.com/science/article/pii/S0888327020307846

[36] https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained

[37]https://docs.google.com/document/d/15ifownZ2LlLXZn0rHlvehu5IOgcHCSYYxQDHj8vZo5g/edit

[38] https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/

[39] https://www.geeksforgeeks.org/ml-xgboost-extreme-gradient-boosting/

4.5 APPENDIX

CODE:

```
from google.colab import drive

drive.mount('/content/drive')

import matplotlib.pyplot as plt

%matplotlib inline

import librosa

import librosa.display

import IPython.display as ipd
```

```
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation
from keras.optimizers import Adam
from keras.utils import to_categorical
from sklearn.preprocessing import LabelEncoder
import os
import pandas as pd
# Function to extract label from file path
def extract_label(file_path):
    # Assuming the label is the last part of the path before the file extension
    label = os.path.splitext(os.path.basename(file_path))[0]
    return label
# Function to create DataFrame with paths and labels
def create_dataframe(audio_folder):
    df_data = {'file_path': [], 'label': []}
    for root, dirs, files in os.walk(audio_folder):
```

```
        for file in files:
            if file.endswith(".wav"):
                file_path = os.path.join(root, file)
                label = extract_label(file_path)
                df_data['file_path'].append(file_path)
                df_data['label'].append(label)
    df = pd.DataFrame(df_data)
    return df
# Specify the folder containing the audio files
audio_folder = '/content/drive/MyDrive/BRagaP'
# Create the DataFrame
df = create_dataframe(audio_folder)
df['label'] = df['label'].str.replace('\d+', '', regex=True)
# Display the resulting DataFrame
print(df)
import os
import pandas as pd
```

```
# Function to extract label from file path

def extract_label(file_path):

    # Assuming the label is the last part of the path before the file extension

    label = os.path.splitext(os.path.basename(file_path))[0]

    return label

# Function to create DataFrame with paths and labels

def create_dataframe(audio_folder):

    df_data = {'file_path': [], 'label': []}

    for root, dirs, files in os.walk(audio_folder):

        for file in files:

            if file.endswith(".wav"):

                file_path = os.path.join(root, file)

                label = extract_label(file_path)



                df_data['file_path'].append(file_path)

                df_data['label'].append(label)

    df = pd.DataFrame(df_data)
```

```
    return df

# Specify the folder containing the audio files

audio_folder = '/content/drive/MyDrive/BRagaP'

# Create the DataFrame

df = create_dataframe(audio_folder)

df['label'] = df['label'].str.replace('\d+', '', regex=True)

# Display the resulting DataFrame

print(df)

df.to_excel('xyz.xlsx')

#import scipy.io.wavfile as wav

#audio1= "/content/drive/MyDrive/BRagaP/bageshree01.wav"

#from scipy.io import wavfile as wav

#wavesr2, wave_audio=wav.read(audio1)

#y, sr = librosa.load(audio1)

#chroma = librosa.feature.chroma_cqt(y=y, sr=sr)

#plt.figure(figsize=(10, 4))

#librosa.display.specshow(chroma, y_axis='chroma', x_axis='time')
```

```
#plt.colorbar()

#plt.title('Chroma Feature')

#plt.show()

#chroma.shape

# ## will use different chroma features

#shifted_data, *other_values = librosa.effects.pitch_shift(y, sr=sr, n_steps=3.0)

# # Get the pitch with the maximum magnitude for each frame

#pitch = pitches[np.argmax(magnitudes, axis=0)]

# # Plot the pitch values over time

#plt.figure(figsize=(12, 4))

#librosa.display.waveshow(y, sr=sr, alpha=0.5)

#plt.plot(librosa.times_like(pitch), pitch, label='Pitch (Hz)', color='r')

#plt.title('Pitch Estimation over Time')

#plt.xlabel('Time (s)')

#plt.ylabel('Pitch (Hz)')

#plt.legend()

#plt.show()
```

```
# pitches, magnitudes = librosa.piptrack(y=y, sr=sr)

# # Get the pitch with the maximum magnitude for each frame

# pitch = np.argmax(magnitudes, axis=0)

# # Plot the pitch values

# # plt.figure(figsize=(10, 4))

# # # librosa.display.specshow(pitch, y_axis='chroma', x_axis='time')

# # plt.colorbar()

# # plt.title('Pitch Estimation')

# # plt.show()

# pitch.shape

# # zero_crossings = librosa.feature.zero_crossing_rate(y)

# # zero_crossings.shape

# # rolloff = librosa.feature.spectral_rolloff(y=y)

# # rolloff.shape

# # spectral_flux = librosa.onset.onset_strength(y=y, sr=sr, feature='spectral')

# # spectral_flux

def features_extractor_mfccs(file):
```

```
    audio, sample_rate = librosa.load(file)
    mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate,n_mfcc= 40);
    mfccs_scaled_features = np.mean(mfccs.T, axis=0)
 #   zero_crossings = librosa.feature.zero_crossing_rate(audio)
 #   energy = librosa.feature.rms(audio)
 #   rolloff = librosa.feature.spectral_rolloff(audio)
 #   spectral_flux = librosa.onset.onset_strength(y=y, sr=sr, feature='spectral')
    return mfccs_scaled_features
extracted_features = []
# Loop through each file
for i in range(len(df)):
#     file_path = os.path.join(folder_path, file_name)
    # Extract features and append to the list
    file_path = df.at[i, 'file_path']
    features = features_extractor_mfccs(file_path)
    file_name = df.at[i, 'label']
#     file_name = file_name.title()
```

```
    extracted_features.append([features, file_name])

# Print the extracted features

# print(extracted_features)

features_df=pd.DataFrame(extracted_features,columns=['feature','class'])

X=np.array(features_df['feature'].tolist())

y=np.array(features_df['class'].tolist())

le=LabelEncoder()

ti=le.fit_transform(y)

y=to_categorical(ti)

tt = np.argmax(y,axis=-1)

tt

ttt = le.inverse_transform(tt)

ttt

features_df

 y.shape

originallabels = le.classes_

print("Original labels: ", originallabels)
```

```
print("Encoded labels: ", np.unique(ti))

from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=127)

X_train_flat = np.array([x.flatten() for x in X_train])

X_test_flat = np.array([x.flatten() for x in X_test])

X_train_flat.shape

print(X_train[0])

print(X_train_flat[0])

num_labels = y.shape[1]

#print(num_labels)

y_train

from tensorflow.keras import layers,models

model = models.Sequential()

model.add(layers.Conv1D(32, kernel_size=5, activation='relu',
input_shape=(X_train_flat.shape[1], 1)))

model.add(layers.MaxPooling1D(pool_size=1))

model.add(layers.Conv1D(64, kernel_size=5, activation='relu'))
```

```
model.add(layers.MaxPooling1D(pool_size=1))

model.add(layers.Flatten())

model.add(layers.Dense(128, activation='relu'))

model.add(layers.Dropout(0.5))

model.add(layers.Dense(num_labels, activation='softmax'))

model.compile(optimizer='adam',

        loss='categorical_crossentropy',

        metrics=['accuracy'])

model.compile(loss='categorical_crossentropy',metrics=['accuracy'],optimizer='Adam')

from keras.callbacks import ModelCheckpoint

from datetime import datetime

num_epochs = 100

num_batch_size = 4

checkpointer                                                                    =
ModelCheckpoint(filepath='saved_model.hdf5',verbose=1,save_best_only=True)

start = datetime.now()
```

```
history_1 = model.fit(X_train, y_train, batch_size=num_batch_size, epochs=num_epochs,
validation_data=(X_test, y_test),  verbose=1,callbacks=[checkpointer])

duration = datetime.now() - start

print("Training completed in time: ", duration)

y_train



plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)

plt.plot(history_1.history['accuracy'], label='Training Accuracy')

plt.plot(history_1.history['val_accuracy'], label='Validation Accuracy')

plt.xlabel('Epochs')

plt.ylabel('Accuracy')

plt.title('Accuracy vs Epochs')

plt.legend()

# Plot loss vs epochs

plt.subplot(1, 2, 2)

plt.plot(history_1.history['loss'], label='Training Loss')
```

```
plt.plot(history_1.history['val_loss'], label='Validation Loss')

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.title('Loss vs Epochs')

plt.legend()

plt.show()

test_accuracy=model.evaluate(X_test,y_test,verbose=0)

print(f"test set accuracy: {test_accuracy[1] * 100}%")



train_accuracy=model.evaluate(X_train,y_train,verbose=0)

print(f"train set accuracy: {train_accuracy[1] * 100}%")

test_accuracy

#predicted = model.predict(X_test_flat)

#tt = np.argmax(predicted,axis=-1)

#tt

#ttt = le.inverse_transform(tt)ttt

#hh = np.argmax(y_test,axis=-1)
```

```
#hhh = le.inverse_transform(tt)

#yhhh

file ="/content/drive/MyDrive/test/bhaitestravi.wav"

x,sr1 = librosa.load(file)

ipd.Audio(x,rate=sr1)

prediction_feature = features_extractor_mfccs(file)

prediction_feature = prediction_feature.reshape(1,-1)

predicted_probabilities = model.predict(prediction_feature)

predicted_class_label = np.argmax(predicted_probabilities)

predicted_class_label = np.array([predicted_class_label])

prediction_class = le.inverse_transform(predicted_class_label)

print("Predicted class:", prediction_class[0])

from sklearn.metrics import confusion_matrix

import seaborn as sns

from sklearn.metrics import confusion_matrix

# confusion matrix

import seaborn as sns
```

```
# Predict the values from the validation dataset

Y_pred = model.predict(X_test)

# Convert predictions classes to one hot vectors

Y_pred_classes = np.argmax(Y_pred,axis = 1)

# Convert validation observations to one hot vectors

Y_true = np.argmax(y_test,axis = 1)

# compute the confusion matrix

confusion_mtx = confusion_matrix(Y_true, Y_pred_classes)

# plot the confusion matrix

f,ax = plt.subplots(figsize=(8, 8))

sns.heatmap(confusion_mtx,                                    annot=True,

linewidths=0.01,cmap="Blues",linecolor="gray", fmt= '.1f',ax=ax)

plt.xlabel("Predicted Label")

plt.ylabel("True Label")

plt.title("Confusion Matrix")

plt.show()

from sklearn.metrics import f1_score
```

```
from sklearn.metrics import precision_score, recall_score

print(f1_score(Y_true, Y_pred_classes,average='weighted'))

print(precision_score(Y_true, Y_pred_classes,average='weighted', zero_division=1))

print(recall_score(Y_true, Y_pred_classes,average='weighted', zero_division=1))

from sklearn.model_selection import cross_val_score

k=10

cv_result = cross_val_score(X_train,y_train,cv=k) # uses R^2 as score

print('Cross_val Scores: ',cv_result)

print('Cross_val scores average: ',np.sum(cv_result)/k)

svm

from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test=train_test_split(X,tt,test_size=0.2,random_state=127)

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)

y_pred_svc = svc_clf.predict(X_test)
```

```
y_pred_svc.shape

def svc_classifier():

    from sklearn.svm import SVC

    from sklearn.metrics import accuracy_score, f1_score

    from sklearn.metrics import confusion_matrix, classification_report

    from sklearn.metrics import accuracy_score, recall_score

    from sklearn.metrics import roc_curve

    import matplotlib.pyplot as plt

    import numpy

    from sklearn import metrics

    # SVC classifier

    svc_clf = SVC(kernel='linear', probability=True, class_weight="balanced")

    svc_clf.fit(X_train, y_train)

    y_pred_svc = svc_clf.predict(X_test)

    #confmat = pd.DataFrame(confusion_matrix(ytest, y_pred_svc))

    acc = accuracy_score(y_test, y_pred_svc)

    print('Test accuracy: {:.4f}'.format(acc))
```

```
    print('Test F1 score: {:.4f}'.format(f1_score(y_test, y_pred_svc, average='weighted')))
    print('Misclassified images: {} out of {}'.format((y_test != y_pred_svc).sum(), len(y_test)))
    print("SVM Training Score:", svc_clf.score(X_train, y_train))
    print("SVM Testing Score:", svc_clf.score(X_test, y_test))
    #classifiers.append(svc_clf)
    #accuracies['SVC'] = acc
    return y_pred_svc,svc_clf
y_pred_svc,svc_clf = svc_classifier()
print(classification_report(y_test, y_pred_svc))
# make predictions and get probabilities
y_pred = svc_clf.predict(X_test)
y_proba = svc_clf.predict_proba(X_test).max(axis=1)
normalize=True
from sklearn.metrics import confusion_matrix
from seaborn import heatmap
# plot confusion matrix
```

```
plt.figure(figsize=(6, 6))

class_labels = ['Bhairav', 'Marwa', 'Poorvi' ,'Yaman' ,'bageshree']

if normalize:

    # heatmap properties

    hm_kwargs = dict(cmap='Blues', square=True, annot=True, fmt='.2f', cbar=False,

                     xticklabels=class_labels, yticklabels=class_labels)

    confu_matrix = confusion_matrix(y_test, y_pred.astype(int), normalize='true')

    heatmap(data=confu_matrix, **hm_kwargs)

    plt.title('Confusion Matrix for SVM')

    plt.yticks(rotation=0)

    plt.show()

else:

    # heatmap properties

    hm_kwargs = dict(cmap='Blues', square=True, annot=True, fmt='d', cbar=False,

                     xticklabels=class_labels, yticklabels=class_labels)

    confu_matrix = confusion_matrix(ytest, y_pred.astype(int))

    heatmap(data=confu_matrix, **hm_kwargs)
```

```
    plt.title('Confusion Matrix')
    plt.yticks(rotation=0)
    plt.show()
from sklearn.model_selection import cross_val_score
k=10
cv_result = cross_val_score(rf,X_train,y_train,cv=k) # uses R^2 as score
print('Cross_val Scores: ',cv_result)
print('Cross_val scores average: ',np.sum(cv_result)/k)
```

Random Forest

```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier(n_estimators=150,random_state = 3)
rf.fit(X_train,y_train)
print("Train ccuracy of random forest",rf.score(X_train,y_train))
print("Test accuracy of random forest",rf.score(X_test,y_test))
RandomForestClassifier_score=rf.score(X_test,y_test)
y_pred=rf.predict(X_test)
```

```
t_true=y_test

from sklearn.model_selection import cross_val_score

k = 10

cv_result = cross_val_score(rf,X_train,y_train,cv=k) # uses R^2 as score

cv_result_randomforest=np.sum(cv_result)/k

print('Cross_val Scores: ',cv_result)

print('Cross_val scores average: ',np.sum(cv_result)/k)

from sklearn.metrics import classification_report, confusion_matrix

from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(random_state = 4)

rf.fit(X_train,y_train)

y_pred = rf.predict(X_test)

cm = confusion_matrix(y_test,y_pred)

print('Confusion matrix: \n',cm)

print('Classification report: \n',classification_report(y_test,y_pred))

from sklearn.metrics import confusion_matrix

from seaborn import heatmap
```

```
# plot confusion matrix
plt.figure(figsize=(6, 6))
class_labels = ['Bhairav', 'Marwa', 'Poorvi' ,'Yaman' ,'bageshree']
if normalize:
    # heatmap properties
    hm_kwargs = dict(cmap='Blues', square=True, annot=True, fmt='.2f', cbar=False,
                     xticklabels=class_labels, yticklabels=class_labels)
    confu_matrix = confusion_matrix(y_test,y_pred)
    heatmap(data=confu_matrix, **hm_kwargs)
    plt.title('Confusion Matrix RANDOM FOREST')
    plt.yticks(rotation=0)
    plt.show()
else:
    # heatmap properties
    hm_kwargs = dict(cmap='Blues', square=True, annot=True, fmt='d', cbar=False,
                     xticklabels=class_labels, yticklabels=class_labels)
    confu_matrix = confusion_matrix(y_test, y_pred)
```

```
    heatmap(data=confu_matrix, **hm_kwargs)

    plt.title('Confusion Matrix')

    plt.yticks(rotation=0)

    plt.show()
```

XGB

```
!pip install xgboost

from xgboost import XGBClassifier

#svm = SVC()

xgb = XGBClassifier()

#svm.fit(x_train,y_train)

xgb.fit(X_train,y_train)

#svm_test_pred = svm.predict(x_test)

xgb_test_pred = xgb.predict(X_test)

#svm_acc = accuracy_score(y_test,svm_test_pred)

xgb_acc = accuracy_score(y_test,xgb_test_pred)

#Accuracy on test data for ML classifiers

#print("SVM accuracy: ",svm_acc)
```

```
print("XGB accuracy: ",xgb_acc)

from sklearn.model_selection import KFold

# Split the data into k folds

kf = KFold(n_splits=5)

# Iterate over the folds

for train_index, test_index in kf.split(X):

    # Train the model on the training fold

    model.fit(X_train, y_train)

    # Evaluate the model on the test fold

    score = model.score(X_test, y_test)

    # Print the score

    print(score)

from sklearn.metrics import confusion_matrix

from seaborn import heatmap

# plot confusion matrix

plt.figure(figsize=(6, 6))

class_labels = ['Bhairav', 'Marwa', 'Poorvi' ,'Yaman' ,'bageshree']
```

```
if normalize:
    # heatmap properties
    hm_kwargs = dict(cmap='Blues', square=True, annot=True, fmt='.2f', cbar=False,
                     xticklabels=class_labels, yticklabels=class_labels)
    confu_matrix = confusion_matrix(y_test, y_pred, normalize='true')
    heatmap(data=confu_matrix, **hm_kwargs)
    plt.title('Confusion Matrix for XGB CLASSIFIER')
    plt.yticks(rotation=0)
    plt.show()
else:
    # heatmap properties
    hm_kwargs = dict(cmap='Blues', square=True, annot=True, fmt='d', cbar=False,
                     xticklabels=class_labels, yticklabels=class_labels)
    confu_matrix = confusion_matrix(ytest, y_pred.astype(int))
    heatmap(data=confu_matrix, **hm_kwa
    plt.title('Confusion Matrix')
    plt.yticks(rotation=0)
```

```
    plt.show()

model.compile(optimizer='adam', loss='mean_squared_error', metrics=['accuracy'])

model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_test, y_test))

test_loss, test_accuracy = model.evaluate(X_test, y_test)

print(f"Test Loss: {test_loss:.4f}, Test Accuracy: {test_accuracy:.4f}")
```