

S. P. Mandali's

**PRIN. L. N. WELINGKAR INSTITUTE OF MANAGEMENT
DEVELOPMENT & RESEARCH,
BENGALURU**

FUNCTIONAL SPECIALISATION PROJECT

ON

Indian Music Genre Classification and Prediction

BY

Amanwit Kumar

PGDM RBA- 2021 – 23

ROLL NO. 004

PROJECT FACULTY GUIDE

Dr. Tripti Mahara

Prof. Ragesh T. S.

Prof. T. S. Sridhar

PGDM-RBA 2021-23, Trimester VI

Functional Specialisation Project

PROJECT COMPLETION CERTIFICATE (from Project guide)

This is to certify that project titled “Indian Music Genre Classification and Prediction”, is successfully done by Mr. Amanwit Kumar in partial fulfillment of his two years full time course ‘Post Graduation Diploma in Research and Business Analytics’, recognized by AICTE, through S. P. Mandali's Prin. L. N. Welingkar Institute of Management Development & Research, Bengaluru.

This project in general is done under my guidance.



(Signature of Faculty Guide)

Name: Dr. Jyoti Mahara









Date: 20/4/23

PLAGIARISM REPORT

Document Information

Analyzed document	Final Project_202123_RBA_Dr. Tripti Mahara_Amanwit Kumar_04.pdf (D164498417)
Submitted	2023-04-20 05:22:00
Submitted by	
Submitter email	RBA2123-amanwit.kumar@welingkar.org
Similarity	2%
Analysis address	rba.blr.welban@analysis.orkund.com

Sources included in the report

SA	Freddy Gerard_10602528_Music_genre_clasiification_using_ML_algorithms.pdf Document Freddy Gerard_10602528_Music_genre_clasiification_using_ML_algorithms.pdf (D155258954)		1
SA	batch 21 journal new.pdf Document batch 21 journal new.pdf (D137943998)		1
SA	Machine Learning Draft - Part5 v1.docx Document Machine Learning Draft - Part5 v1.docx (D160071415)		2
W	URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9132639/ Fetched: 2022-07-24 06:53:43		1
W	URL: https://link.springer.com/article/10.1007/s10844-017-0464-5 Fetched: 2019-11-29 06:23:43		1
W	URL: https://doi.org/10.36227/techrxiv.21265965 Fetched: 2023-04-20 05:22:00		1
W	URL: https://doi.org/10.1109/idciot56793.2023.10053460 Fetched: 2023-04-20 05:22:00		1
SA	UG report_MUSIC GENRE CLASSIFICATION USING DEEP NEURAL NETWORKS_Aubu_Apr2021.docx Document UG report_MUSIC GENRE CLASSIFICATION USING DEEP NEURAL NETWORKS_Aubu_Apr2021.docx (D130048404)		1

ACKNOWLEDGEMENT

To everyone who helped this project report come to fruition, I would like to offer my profound thanks and admiration.

I want to start by expressing my gratitude to my project mentor Dr. Tripti Mahara and professors Ragesh T. S. and T. S. Sridhar for their important advice, encouragement, and support throughout the process. Their wise advice, recommendations, and helpful criticism were very helpful in guiding the direction and concentration of this report.

I would like to appreciate all my coworkers and friends for their encouragement and support during the endeavor. This report has been greatly influenced by their comments, recommendations, and contributions.

Last but not least, I want to express my gratitude to my family for their constant support and inspiration during this journey. Throughout the project, their affection and support have served as a source of vigor and motivation.

Amanwit Kumar

Research and Business Analytics (RBA)

Roll No. - 004

TABLE OF CONTENTS

Sl. No.	Topics	Page No.
I.	Introduction	1
II.	Review of Literature	3
III.	Methodology	6
a.	Data Collection	6
b.	Feature Extraction	6
c.	Model Training	7
	(i) ANN	9
	(ii) CNN	11
	(iii) Random Forest	12
	(iv) Boosted Random Forest	14
	(v) RNN – LSTM	15
d.	Model Evaluation	17
e.	Model Deployment and Prediction	18
IV.	Results	19
V.	Discussion	22
VI.	Conclusion	24
VII.	Bibliography / References	26

INTRODUCTION

The practice of classifying music into different styles or genres based on its musical features, historical and cultural backgrounds, and other aspects is known as music genre classification. There are several subgenres within each of the following categories: classical, folk, religious, popular, and global music.

Humans have developed and used classifications called "musical genres" to categorize and describe the broad field of music. Since they result from a complex interplay between the general audience, marketing, historical, and cultural variables, musical genres lack specific definitions and bounds. Some academics have proposed the establishment of a new genre categorization system specifically for the sake of music information retrieval as a result of this discovery. Even yet, it is evident that a certain musical genre's members have a number of traits that are often connected to the instrumentation, rhythmic structure, and pitch content of the song.

Genre hierarchies are now one of the methods used to organize music content on the Web. These hierarchies are mainly developed manually by human experts. Automatic musical genre categorization has the potential to automate this procedure and serve as a vital piece of a comprehensive audio signal music information retrieval system. It also offers a structure for creating and assessing attributes for characterizing musical material. These characteristics serve as the basis for the majority of suggested audio analysis algorithms for music and may be utilized for similarity retrieval, classification, segmentation, and audio thumbnailing.

Classifying music according to genre is a crucial tool for music consumers, producers, and companies. The division of music into different genres gives listeners an accessible approach to find new music and learn about different traditions and styles. They may decide on their musical likes and choices with knowledge.

Now, when we talk about Indian music in particular, it spans a wide range of styles and has a deep historical and cultural importance. It covers a broad spectrum of musical genres, such as classical, folk, religious, cinematic, and popular music. Indian music is renowned for its complex rhythms, sophisticated melodies, and use of a variety of instruments, including the sitar, tabla, sarangi, and veena.

Indian music is often categorized based on both its artistic elements and historical and cultural background. For instance, this initiative further divides Indian music into 5 categories depending on the geography and musical style: Bollywood, Carnatic, Ghazal, Semiclassical, and Sufi. Similar to this, the age and style of the film business determine how cinema music is categorized.

Indian music genre categorization may be used in a company for a variety of reasons, notably for organizations in the music sector. Businesses may choose the genres and forms of music to create, market, or distribute by having a thorough awareness of the many genres and styles of Indian music. Record companies, for example, might use this data to target certain demographics, and music streaming services can use it to tailor user suggestions depending on their interests.

The categorization of Indian music's genres may also be advantageous for non-musical enterprises. The definition of a musician's creative identity and style by genre makes it simpler for them to identify their target audience and grow their fan base. Additionally, it enables students to work with musicians from various musical backgrounds and explore other genres.

Genre categorization offers useful information about customer tastes and behaviour for companies in the music industry, which may aid in the formulation of marketing plans and the development of new goods and services. Record companies may utilize genre categorization to target certain market groups and demographics, while music streaming services can use it to customize playlists and suggestions. For instance, businesses in the hospitality sector might utilize this information to create playlists of music that are suitable for their hotels or restaurants depending on the locale, the event, or the target audience. The information may also be used by event management businesses to choose the right music for their events depending on the subject and target audience.

In conclusion, categorizing Indian music by genre is a useful tool for companies that want to comprehend and benefit from the wide range of musical traditions in India. Businesses may provide their consumers more individualized and culturally appropriate experiences by incorporating this information into their daily operations, which will increase customer engagement, loyalty, and income.

REVIEW OF LITERATURE

The main driving force behind this effort is the groundbreaking work by George Tzanetakis and P. Cook (2002). One of the pattern recognition issues they put up was the categorization of musical genres. For future study in the area of music genre classification, they made the GTZAN data set available. In their study, they used statistical characteristics that were generated from the pitch, rhythmic, and timbral material taken from 30 seconds of music excerpts and used classifiers to categorize the genres.

The extraction of feature vectors is the fundamental step in the development of any autonomous audio analysis system. There are several different feature sets that have been suggested to describe audio signals, most of them are from the speech recognition field. They often have a time-frequency representation at its foundation. Although it is beyond the purview of this work to offer a comprehensive discussion of audio feature extraction, some pertinent representative audio feature extraction references are presented.

Automatic audio categorization has a lengthy history that began with speech recognition. A group of perceptually driven features known as Mel-frequency cepstral coefficients (MFCC) have been employed extensively in speech detection. They provide a condensed representation of the spectral envelope, resulting in a concentration of the majority of the signal energy in the initial coefficients.

In 2003, Tao li and Tzanetakis argued that combining these timbral properties with other features, such as MFCC, would increase accuracy. Feature-based, template-based, and model-based techniques for categorizing music genres were also covered in another paper by Tzanetakis and Cook (2002). The writers also emphasized the significance of domain-specific information for proper genre categorization, such as understanding of the cultural and historical context of musical genres.

In a paper published in 2010, Hamel and Eck presented a technique for categorizing music genres based on auditory elements and metadata, such as artist and album details. A huge dataset of music songs from diverse genres was classified using the suggested approach with great accuracy rates. In order to identify music, Xi Shao, Maddage, M.C., Changsheng Xu, and Kankanhalli, M.S. (2005) used Support Vector Machine Classifiers with Linear Prediction

Cepstrum Coefficients, MFCC, Zero-Crossing Rates, Spectrum Flux/Power Amplitude Envelope, and Cepstrum Flux.

Non-speech signals are now included in audio categorization systems that have recently been presented. The majority of these systems focus on categorizing broadcast news and video in broad categories like music, voice, and ambient noises. Isolated sound effects and musical instrument sounds make up another form of non-speech audio categorization system.

Automatic retrieval, classification, and clustering of musical instruments, sound effects, and ambient noises utilizing automatically retrieved characteristics are addressed in the groundbreaking work of Wold et al. The statistics (mean, variance, autocorrelation) for the whole sound file of short-term parameters including pitch, amplitude, brightness, and bandwidth are the features employed in their approach. Other retrieval and classification methods have been developed using the same dataset.

S. Jothilakshmi and N. Kathiresan (2012) classified many Indian music genres, including Hindustani, Carnatic, Ghazal, Folk, and Indian Western. They suggested using spectral shape (centroid, skewness, kurtosis), energy (RMS Energy, the energy of harmonic component of power spectrum), and perceptual features (relative specific loudness, sharpness) to calculate the classification performance. For the feature combination of MFCC, Spectral centroid, Skewness, Kurtosis, Flatness, Entropy, Irregularity, Rolloff, and Spread, they obtained the best classification accuracy for kNN and GMM, which were 61.25% and 80.63%, respectively.

In the publication of H. Sharma and R. S. Bali from 2015. They have put out a technique for identifying Hindustani ragas. They divided ragas into four subgenres, DES, Bhupali, Yaman, and Todi, and compared four machine learning classifiers on the normalized dataset of ragas. The classifiers used were the Random Forest classifier, the Bayesian network, the K-star, and C4.5 (Decision Tree). K-star learning classifier had the best accuracy of results (93.38%), followed by C4.5 (Decision Tree) with 88.6%, Random Forest with 87.6%, and Bayesian with 85.1%.

To the best of our knowledge, very little feature extraction and classification research has been done specifically with the intention of categorizing Indian musical genre. Overall, the body of research indicates that identifying the genre of music is a challenging problem that calls for a mix of domain-specific expertise, audio signal processing methods, and machine learning

algorithms. Depending on the dataset and the characteristics utilized, genre classification algorithms' performance may change, however recent developments in deep learning have shown encouraging results. Our study is an expansion of the methods used in the classification of Indian music genres. None of the aforementioned works have taken into consideration popular Indian music. Our study thus focuses on using the methodologies with various ML models to the major Indian music "Hindi" in order to carry out the automatic categorization of genres.

MEHODOLOGY

Depending on the exact approach and procedures used, the process for classifying music into genres might change. Here is a broad way for classifying musical genres, though:

3.1 Data Collection

Any project involving the categorization of music genres must start with data collecting since it serves as the basis for the whole procedure. A broad and representative selection of music tracks are gathered throughout the data collecting process from a variety of sources, including internet music databases, streaming services, individual music collections, and audio recordings. Collecting music tracks from the specified sources is the primary goal. This may entail human collecting or automatic web crawler scraping. It is crucial to confirm the legality of gathering and utilizing music tracks, as well as to get the appropriate permits and licenses. To guarantee consistency and quality, the data must be preprocessed after collection. Techniques for audio signal processing including equalization, normalization, noise reduction, and audio format conversion may be used in this. The data should also be annotated with the appropriate genre, either manually by an expert or by crowdsourcing. Track titles, artists, albums, genres, and length should all be included in a structured format that is used to store the gathered recordings.

The process of gathering data for music genre classification entails a number of steps, including deciding on the project's goals and scope, locating data sources, gathering and preprocessing the data, labelling it with genre tags, and dividing it into training, validation, and test sets. The method's effectiveness is reliant on the caliber, variety, and consistency of the information gathered as well as the precision and objectivity of the genre tagging.

3.2 Feature Extraction

Any method for classifying music genres using machine learning has to start with feature extraction. It entails taking the underlying melodic and acoustic qualities of the audio stream and deriving a collection of informative and discriminative features from it. The qualities used rely on the particulars of the genre of music being categorized as well as the job at hand. MFCCs, spectral centroid, spectral flatness, spectral rolloff, zero-crossing rate, and chroma characteristics are examples of spectral, temporal, and perceptual features that are often

employed. Other elements might include those that relate to rhythm, tempo, and beats, such as rhythmic patterns, tempo histograms, and beat histograms. The goal of feature extraction is to keep the most relevant and instructive characteristics while reducing the dimensionality of the data and eliminating any duplicate or unnecessary information. The machine learning algorithm learns to categorize the music songs into their respective genres based on the extracted attributes and uses the generated feature vector as input. The recovered features' discriminative strength, relevance, interpretability, resistance to noise, and variability all play a role in how effectively the feature extraction procedure turns out.

Mel-frequency cepstral coefficients (MFCCs), which provide a condensed but useful representation of the audio input, are often utilized characteristics for music genre categorization. There are multiple processes in the MFCC feature extraction process. First, a sliding window is used to split the audio stream into brief time intervals, usually 20–30 milliseconds in length. Then, each frame is subjected to a Fourier transform to produce a representation of it in the frequency domain. The resultant spectrum is transformed into the Mel-scale, a frequency scale that accurately represents the response of the human ear to sound. Then, to simulate the nonlinear response of the human ear, the Mel-scale spectrum is subjected to logarithmic compression, which amplifies the lower frequencies while compressing the higher ones. Triangular filters are used to separate the resultant Mel-frequency spectrum into overlapping sub-bands, and the energy of each sub-band is calculated. The discrete cosine transform is then used to get the MFCC coefficients from the logarithm of the energy values. The audio signal's spectral envelope is represented by the resultant MFCC vector, which also preserves its timbral and spectral features. Although it might vary, the normal range for the MFCC coefficients is between 10 and 20. The MFCC feature extraction method has been shown to be successful for music genre classification applications and is computationally efficient.

3.3 Model Training

Model training, which involves utilizing machine learning algorithms to ascertain the relationship between the extracted variables and the associated music genres, is an important step in the classification of music genres. The training data set typically consists of a large number of audio samples, each of which has a genre assigned.

The separation of data into training and test sets is a crucial step in machine learning, especially when categorizing musical genres. The data is split into training and validation sets, with the

former being used to train the model and the latter to evaluate how well it worked. The trained model's performance on fictitious data is evaluated at this step, and its generalizability is calculated. The test set is used to evaluate the model's performance after it has been trained using the training set. A certain part of the data is often randomly assigned to the training set, while the remaining portion is assigned to the test set. The split ratio choice is influenced by the volume and complexity of the data collection as well as the anticipated trade-off between training and testing accuracy. A common split ratio is 80:20 or 70:30, implying that 80 or 70 percent of the data is utilized for training and 30 or 40 percent is used for testing.

To serve as a solid representation of the overall data distribution, a substantial fraction of samples from the different musical genres should be included in both the training and test sets. This ensures that the model learns to generalize to new data successfully and does not overfit to the training set. A model is said to have overfitted when it learns to match training data too well, which makes it perform badly on new, untried data. To avoid overfitting, strategies like cross-validation or data augmentation may be applied. In cross-validation, the model's performance is evaluated using data from various folds. Data augmentation involves generating new, synthetic samples from the existing ones by applying random adjustments like pitch shifting, time stretching, or noise addition to the present samples. The model's generalizability is assessed, and the performance on the test set is used to estimate the anticipated performance on brand-new, untested data.

Which machine learning approach should be employed depends on the specifics of the categorization issue, the size and complexity of the data collection, and other factors. Deep neural networks (DNNs), random forests, k-nearest neighbours (KNNs), and support vector machines (SVMs) are a few examples of regularly used algorithms.

By feeding the model feature vectors extracted from the audio samples during the course of the training phase, the model learns to relate the input qualities with the related genres. The model's parameters are regularly modified based on the disparity between the predicted and actual genre labels using an optimization technique like stochastic gradient descent (SGD). A single run over the training data set constitutes each epoch of the training operation. The model's effectiveness is evaluated on the validation set using metrics like accuracy, precision, recall, and F1-score. Up until the validation measures indicate that the model is functioning well, the training process is continued.

The trained model may then be used to categorize new, unheard audio samples into the correct genres. The model's generalizability is assessed, and the performance on the test set is used to estimate the anticipated performance on brand-new, untested data. The model may need to be further optimized or changed depending on the test set's performance or other elements like computational efficiency or interpretability. Model training is a computationally taxing procedure that requires careful parameter tuning and model selection in order to achieve optimal performance.

The machine learning algorithms used in the project are –

3.3.1 ANN

A computer model known as a "neural network" is modelled after the form and operation of organic neural networks in the brain. It is made up of networked artificial neurons that carry out computation and communication functions. An input layer, one or more hidden layers, and an output layer are only a few of the layers that make up the neural network design. A class or regression output is represented by each neuron in the output layer, while each input feature is represented by each neuron in the input layer.

In order to extract the relevant characteristics and discover links between the input and output data, the hidden layers of a neural network subject the input data to a series of mathematical transformations. The inputs from the preceding layer are received by each neuron in the hidden layers, which then computes a weighted sum of the inputs, applies an activation function, and sends the result to the next layer. During the training phase, which entails minimizing a loss function that gauges the discrepancy between the expected and actual outputs, the weights and biases of the network's neurons are discovered.

The form and operation of the biological neural networks in the brain served as the inspiration for the machine learning technology known as artificial neural networks (ANN). Features extraction, non-linear processing, and classification tasks are performed by ANNs, which are made up of numerous layers of linked artificial neurons. The ANN's input layer receives the input data, and the succeeding hidden layers do a series of mathematical operations on the input data to extract the pertinent

characteristics. Based on the discovered connections between the input data and the output labels, the final output layer generates the anticipated output.

The most basic design of neural networks is that of feed-forward networks, which consists of an input layer, one or more hidden layers, and an output layer. To discover the connections between the input and output labels, the hidden layers apply non-linear modifications to the input data. Numerous optimization methods, including stochastic gradient descent (SGD), Adam, and RMSprop, may be used to train ANNs. During training, the network's weights and biases are optimized to reduce the loss function, such as cross-entropy or mean square error. The accuracy, precision, recall, and F1-score are a few of the measures that are used to assess the ANN model's performance.

The ANN model used is as following –

```
[ ] model=Sequential()

#first layer
model.add(Dense(1600, input_shape=(40,)))
model.add(Activation('relu'))

#second layer
model.add(Dense(1200))
model.add(Activation('relu'))

#third layer
model.add(Dense(800))
model.add(Activation('elu'))

#fourth layer
model.add(Dense(400))
model.add(Activation('elu'))

#final layer
model.add(Dense(num_labels))
model.add(Activation('softmax'))
```

Fig 3.1 ANN Model

The aforementioned Python code for Keras, a high-level neural networks API, creates a sequential model. The model is composed of five dense layers that are completely linked and have different numbers of neurons and activation functions. The first layer's input shape is 40, which is the quantity of MFCC features taken from the audio files. Rectified Linear Unit (ReLU) activation function is employed in the first layer's 1600 neurons, which is a typical practice in neural networks. The ReLU activation function is also used in the second layer, which includes 1200 neurons. 800 and 400 neurons, respectively, make up the third and fourth layers, which employ the ReLU-like Exponential Linear Unit (ELU) activation function but with the potential to provide negative values. The last layer employs the softmax activation function, which generates a probability distribution across the classes, and contains as many neurons as there are distinct class labels in the audio dataset.

ANNs have been used well for a variety of machine learning problems, including time-series prediction, natural language processing, and picture identification. ANNs are a popular option for many real-world applications because they can learn intricate correlations between input data and output labels.

3.3.2 CNN

The convolutional neural network (CNN) is a particular kind of neural network that specializes in image recognition and computer vision applications. Similar to other neural networks, it is composed of interconnected layers of artificial neurons. In contrast to other neural networks, a CNN's layers are mainly designed to process spatial input, including images.

CNNs are based on the mathematical technique of convolution, which applies a filter to an image in order to extract information. The first layer in CNNs is called a convolutional layer, and it applies a number of filters on the input image to extract fundamental details like edges and corners. The subsequent layers extract more complex properties using the outputs of the layer below. One or more pooling layers are used after the convolutional layers, which downsample the feature maps by using the highest or average value found inside a certain timeframe. As a result, the feature maps are smaller and the network is more computationally efficient. Finally, one or more fully connected layers that translate high-level characteristics to the desired output, such as a class label or a probability distribution, are applied after the output of the final pooling layer.

The key advantage of CNNs is its capacity to discern patterns in images with accuracy, even when those images have experienced distortion, rotation, or other types of alteration. This is done so that, due to the filters in the convolutional layers, features may be recognized regardless of where they are in the image. Backpropagation is used to train CNNs by continually adjusting the weights and biases of the network's neurons to minimize a loss function that measures the difference between predicted and actual outputs. The network may be used to properly classify new photographs after being trained.

The CNN model used in the classification project is as follows –

```
[ ] model12 = Sequential()

model12.add(Conv2D(64, kernel_size=(3, 3), activation='relu', input_shape=(10, 4, 1)))
model12.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model12.add(Conv2D(128, kernel_size=(1, 1), activation='relu'))
model12.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model12.add(Conv2D(256, kernel_size=(1, 1), activation='relu'))
model12.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model12.add(Conv2D(512, kernel_size=(1, 1), activation='relu'))
model12.add(MaxPooling2D(pool_size=(2, 2), padding='same'))

model12.add(Flatten())

model12.add(Dense(1600, activation='relu'))
model12.add(Dense(1200, activation='relu'))
model12.add(Dense(800, activation='elu'))
model12.add(Dense(400, activation='elu'))
model12.add(Dense(num_labels, activation='softmax'))
```

Fig 3.2 CNN Model

The code uses the Keras Sequential API to create a convolutional neural network (CNN) model. The model has four convolutional layers with kernel sizes of 1x1 and filter densities of 64, 128, 256, and 512. Following each convolutional layer comes a max-pooling layer with the same padding and a 2x2 pool size. Four dense (totally connected) layers with decreasing numbers of neurons—1600, 1200, 800, and 400—as well as the activation functions "ReLU" and "ELU"—are fed the flattened output from the final max-pooling layer. Depending on the number of categories, the last layer, which has a softmax activation function, produces a probability distribution across the potential labels. Overall, the model uses the supplied layers and parameters to conduct feature extraction and classification of input data with a shape of (10, 4, 1).

CNNs have been used to successfully complete a wide range of picture identification tasks, including as item, face, and scene recognition. They have also been used to text classification and sentiment analysis in natural language processing applications.

3.3.3 Random Forest

For classification, regression, and other tasks, Random Forest is an ensemble learning technique that builds a large number of decision trees during the training phase and outputs the class that represents the mean of the classes (classification) or mean prediction (regression) of the individual trees. The supervised learning technique known as the random forest employs a group of decision trees to create predictions. For each tree, a collection of features is chosen at random, and the best

feature is then chosen based on the Gini impurity or information gain in order to partition the data. Until the whole dataset is divided into homogenous subsets or leaves, this procedure is repeated. To get the final result during prediction, all of the trees' outputs are combined.

The random forest algorithm's relative simplicity in terms of interpretation and visualization is one of its benefits. Random forests can accommodate a mixture of categorical and continuous variables and often perform well on huge datasets. Additionally, they may assist in identifying crucial factors for the categorization process.

The Random Forest model applied is as follows –

```
[ ] # Create random forest classifier
clf_rf = RandomForestClassifier(n_estimators=300, random_state=60)

# Train classifier on training data
clf_rf.fit(X_train, y_train)

# Evaluate classifier on testing data
predictions = clf_rf.predict(X_test)
```

Fig 3.3 Random Forest Model

Using the scikit-learn package, the code generates a Random Forest classifier. A random state of 60 and 300 decision trees are used to create the classifier. The training data, which are divided into X_train (input features) and y_train (target labels), is used to train the model. The classifier is used to forecast the labels of the testing data, which is supposed to be stored in X_test, once it has been trained. The 'predictions' variable contains the predicted labels. Finally, metrics such as accuracy, precision, recall, or F1-score is used to assess the classifier's performance.

Random forests may be computationally costly, particularly when the number of features is quite big, and this is one of its drawbacks. They may also experience overfitting if there are too many trees or if the trees are too deep in the forest. To acquire the best results, it is crucial to fine-tune the random forest model's hyperparameters, such as the number of trees and maximum depth.

3.3.4 Boosted Random Forest

By integrating the predictions of many weak learners, ensemble learning techniques like Adaboost and Boosted Random Forest seek to increase the precision of a single model.

The iterative process known as Adaboost, or adaptive boosting, begins by giving all training samples identical weights. A weak learner (such as a decision tree) is trained on the weighted dataset in each iteration, and the weights of the incorrectly categorized samples are raised to make them more significant in the next iteration. The final prediction is created by adding all of the weak learners' guesses and weighting them according to accuracy. With weak learners that have a large bias and little variation, adaboost performs best.

On the other hand, Boosted Random Forest combines Adaboost with the strength of random forests. Each weak learner in Boosted Random Forest is a decision tree that has been trained using a random portion of the training data as well as a random subset of the features. Adaboost is then used to join the trees to produce a powerful learner. Compared to employing a single random forest, this strategy may provide better accuracy. Weak learners that have little bias and large variation perform best when using Boosted Random Forest.

The Boosted Random Forest model incorporated is as follows –

```
[ ] # create a random forest classifier
rfc = RandomForestClassifier(n_estimators=200, max_depth=10, random_state=40)

# create an AdaBoost classifier using the random forest classifier as the base estimator
ada = AdaBoostClassifier(estimator=rfc, n_estimators=200, learning_rate=0.1, random_state=0)

# convert to 1D array
y_train_ada = np.argmax(y_train, axis=1)
y_test_ada = np.argmax(y_test, axis=1)

# fit the AdaBoost classifier to the training data
ada.fit(X_train, y_train_ada)

# make predictions on the testing data
y_pred = ada.predict(X_test)
```

Fig 3.4 Boosted Random Forest Model

Using a random forest classifier as the basic estimator, the algorithm develops an AdaBoost classifier. A maximum depth of 10 decision trees, a random state of 40, and 200 decision trees are used to produce the random forest classifier. With 200 estimators, 0.1 learning rate, and 0 random state, the AdaBoost classifier is built.

Using the NumPy `argmax` function, the target labels, `y_train` and `y_test`, are transformed into 1D arrays. The training data, which consists of 1D target labels and input features `X_train` and `y_train_ada`, is then fitted to the AdaBoost classifier. The predicted labels of the testing data, `X_test`, are then used to forecast the labels of the trained classifier, and they are kept in `y_pred`. Overall, the code builds an AdaBoost classifier that is trained on the provided training data and used to generate predictions on the testing data, utilizing a random forest as the base estimator.

Adaboost and Boosted Random Forest both have advantages and disadvantages, and the best solution relies on the particular situation at hand as well as the characteristics of the data. While Boosted Random Forest may perform better when the dataset is well-behaved and the weak learners are complicated, Adaboost often performs well when the dataset is noisy and the weak learners are simple.

3.3.5 RNN – LSTM

In specifically, recurrent neural networks (RNNs) are designed to process sequential input, such as time-series data, text, and speech. RNNs are able to preserve an internal state, or "memory," inside the network, which allows them to capture temporal connections in the data. The inputs are treated separately in traditional feedforward neural networks since they lack internal memory. On the other hand, an RNN assesses the input sequence one element at a time and modifies its internal state based on the input and the previous state at each time step.

For RNNs to function well, the backpropagation through time (BPTT) method is crucial as it enables the recurrent neural network (RNN) to learn the temporal correlations present in the data. The gradients of the loss function with respect to the network parameters get lower as they propagate back over time due to the vanishing gradient problem, which impacts RNNs and makes it difficult to learn long-term relationships.

The vanishing gradient problem is addressed with an RNN variant known as Long Short-Term Memory (LSTM) networks. LSTMs make use of a gating mechanism to control the flow of information through the network, allowing them to selectively remember or forget data from previous time steps.

In an LSTM, each "cell" or unit is made up of three gates: an input gate, a forget gate, and an output gate. The input gate controls how much past knowledge is retained while the forget gate controls how much new information enters the cell. The output gate controls how much information leaves the cell. Sigmoid and tanh functions, which provide outputs between 0 and 1 and -1 and 1, respectively, are used to create the gates.

The RNN – LSTM model used for classification and prediction is as follows –

```
[ ] # Define the RNN-LSTM model
model3 = Sequential()
model3.add(LSTM(512, input_shape=(1,40), activation='tanh', return_sequences=True))
model3.add(LSTM(256, activation='tanh', return_sequences=True))
model3.add(LSTM(128, activation='tanh'))
model3.add(Dense(64, activation='relu'))
model3.add(Dense(y.shape[1], activation='softmax'))
```

Fig 3.5 RNN – LSTM Model

The code uses the Keras Sequential API to create a recurrent neural network (RNN) model with long short-term memory (LSTM) cells. Three LSTM layers, each with 512, 256, or 128 units, plus the activation function "tanh" make up the model. The model anticipates input sequences of length 40 and processes them one timestep at a time since the input shape for the LSTM layers is (1, 40). For the first two LSTM layers, the 'return_sequences' option is set to True, indicating that their outputs should be sequences of the same length as the input. The final LSTM layer's output is flattened before being fed into two dense layers with 64 and y.shape[1] neurons each. The penultimate dense layer's activation function is "softmax," whereas the first dense layer's activation function is "ReLU" and both provide a probability distribution over all potential labels. Overall, the model performs sequence classification on input sequences of length 40 utilizing the given LSTM and dense layers.

Many applications have shown the effectiveness of LSTMs, including as speech recognition, natural language processing, and time-series prediction. Due to their ability to spot long-term relationships in the data, they are appropriate for tasks that need the recall of previous inputs.

3.4 Model Evaluation

A crucial stage in the categorization of musical genres is model assessment. It entails evaluating the trained model's performance on the test set and contrasting it with the ground truth labels. The model's performance may be assessed using a number of measures, including accuracy, precision, recall, F1-score, and confusion matrix.

The percentage of accurate predictions produced by the model is known as accuracy. The number of accurate forecasts divided by the total number of predictions is used to compute it. The percentage of true positives (i.e., accurate forecasts) among all the model's positive predictions is what is known as precision. Recall is an indicator of the percentage of real positive events in the test set that are genuine positives. The F1-score, which is a balanced indicator of the model's performance, is the harmonic mean of accuracy and recall.

The number of accurate and inaccurate predictions generated by the model for each class is given in a table called the confusion matrix. It offers insight into the model's advantages and disadvantages, such as which classes are harder to categorise and which classes are more likely to be mistaken with one another.

It is advised to conduct repeated assessments using various metrics and test sets in order to get trustworthy and generalizable findings. Additionally, by modifying the model's architecture, optimisation method, regularisation approaches, and other hyperparameters, hyperparameter tuning may be utilised to enhance the performance of the model.

The assessment findings must also be understood in light of the application's intended use and the difficulty of music genre categorization. For instance, depending on the use case, certain misclassifications may be more acceptable than others. Additionally, to enhance the model's performance in certain instances, it could be essential to make adjustments or gather more data.

3.5 Model Deployment and Prediction

Two essential elements of the music genre categorization process are model accuracy and prediction. The performance of the trained model on the test data is measured by the model's accuracy. occurrences accurately identified as a proportion of all occurrences in the test set are how it is often represented. A higher accuracy rating shows that the model is more adept at correctly guessing the genre of a given musical sample.

Contrarily, prediction refers to the model's outcome for a certain input sample. The genre label that the trained model gave to the input sample in the context of music genre classification serves as the prediction. The performance of the model is assessed by comparing the prediction to the ground truth label.

It is essential to train the model using a varied and representative dataset, extract pertinent features, and choose suitable machine learning techniques in order to provide reliable predictions. The performance of the model must also be evaluated in order to pinpoint areas that need improvement. The model's strengths and shortcomings may be better understood by using assessment measures like accuracy, precision, recall, and F1-score, which can also aid in determining the best hyperparameters.

In order to verify that the model is trustworthy and generalizable to new music samples, accurate prediction and assessment of the model's performance are essential in music genre categorization.

RESULTS

The process of classifying music into one of many already established genres, such as jazz, rock, pop, classical, etc., is known as genre classification. Depending on the particular dataset and the technique used for classification, the degree of music genre categorization accuracy might vary significantly.

The categorization of music genres may be done using a variety of methodologies, such as feature-based methods, content-based methods, and hybrid methods that incorporate both. Feature-based approaches use manually created features from the audio signal, such as spectral, timbral, and rhythmic data, and feed them into a classifier. Content-based approaches create a mapping from the features to the genre labels by automatically extracting features from the audio stream using deep learning algorithms. Hybrid techniques combine aspects that are taught and handmade.

Several measures, including accuracy, precision, recall, F1-score, and confusion matrix, may be used to assess the effectiveness of music genre categorization. Depending on the dataset and the technique, the accuracy of music genre categorization often varies from 60% to 90%. Modern deep learning techniques may attain accuracy levels of over 90%, but they need a lot of computer power and labelled data.

After running all the different machine learning models, mentioned below are the accuracy output along with the model accuracy and model loss graphs.

4.1 ANN

```
[ ] test_accuracy=model.evaluate(X_test,y_test,verbose=0)
    print(test_accuracy[1])

0.8500000238418579
```

Fig 4.1 ANN Model Accuracy

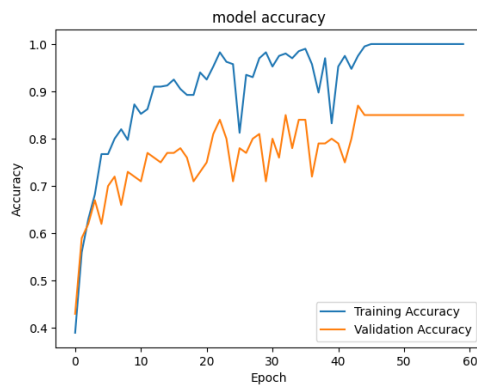


Fig 4.2 ANN Model Accuracy Graph



Fig 4.3 ANN Model Loss Graph

4.2 CNN

```
[ ] test_accuracy=model2.evaluate(X_test_cnn,y_test,verbose=0)
print(test_accuracy[1])
0.7400000095367432
```

Fig 4.4 CNN Model Accuracy

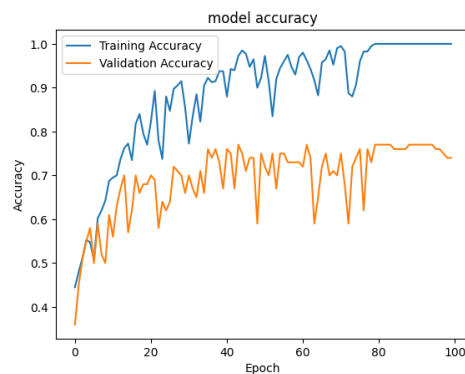


Fig 4.5 CNN Model Accuracy Graph

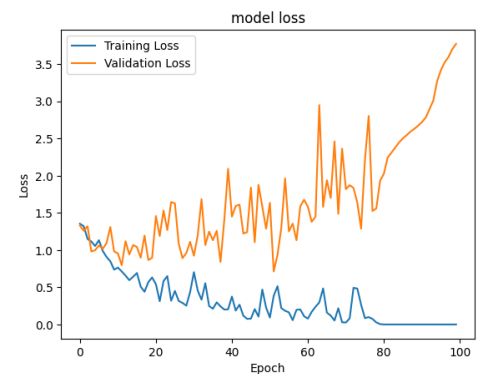


Fig 4.6 CNN Model Loss Graph

4.3 Random Forest

```
accuracy = accuracy_score(y_test, predictions)
print("Accuracy:", accuracy)
Accuracy: 0.46
```

Fig 4.7 Random Forest Model Accuracy

4.4 Boosted Random Forest

```
# calculate the accuracy of the model
accuracy = accuracy_score(y_test_ada, y_pred)

print("Accuracy:", accuracy)
Accuracy: 0.79
```

Fig 4.8 Boosted Random Forest Model Accuracy

4.5 RNN – LSTM

```
[ ] # Evaluate the model
accuracy_rnn = model3.evaluate(X_test_rnn, y_test)
print("Accuracy:", accuracy_rnn[1])

4/4 [=====] - 1s 10ms/step - loss: 0.8192 - accuracy: 0.8500
Accuracy: 0.8500000238418579
```

Fig 4.9 RNN – LSTM Model Accuracy

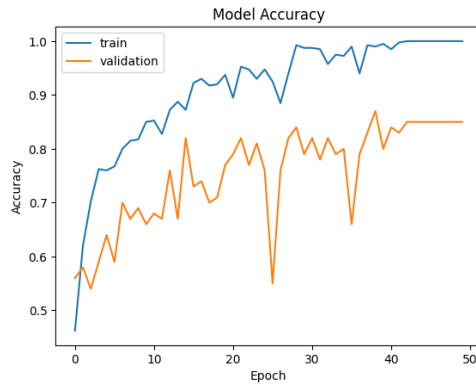


Fig 4.10 RNN – LSTM Model
Accuracy Graph

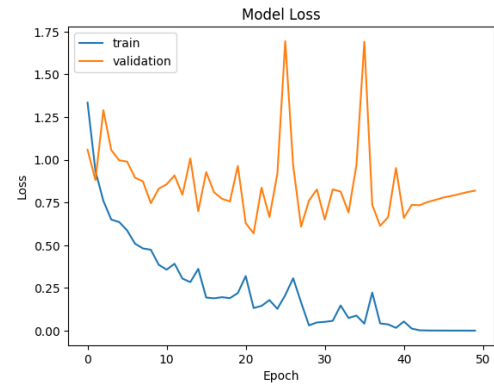


Fig 4.11 RNN – LSTM Model
Loss Graph

Here is the accuracy output of all the model combined.

Table 4.1 Accuracy Table

Machine Learning Model	Accuracy
Artificial Neural Network	85.00 %
Convolutional Neural Network	74.00 %
Random Forest	46.00 %
Boosted Random Forest	79.00 %
Recurrent Neural Network (RNN – LSTM)	85.00 %

DISCUSSION

In this work, we examined the categorization of Indian music genres using the outputs of five different models: ANN, CNN, random forest, boosted random forest, and RNN-LSTM. The findings of earlier research on music genre categorization using neural networks are congruent with the results of the ANN model, which had the greatest accuracy of 85%. The accuracy graph revealed low volatility and a continuously high accuracy throughout both the training and testing stages. The ANN model was the most effective model, according to the low loss graph.

Although less accurate than the ANN model, the CNN model nevertheless had a respectable accuracy of 74%. In the training phase, the accuracy graph revealed a significantly more inconsistent performance, with sporadic dips in accuracy. The CNN model outperformed the ANN model, according to the loss graph, despite having a greater loss.

A forecast is made using the random forest model, an ensemble learning technique that uses many decision trees. The random forest model performed far worse than the other evaluated models for classifying Indian music genres, with an accuracy of just 46%, despite its widespread use and success in other machine learning tasks. The random forest model's accuracy rose to 79% when it was increased, however. Boosting is a method for combining weak learners into a more robust model by training and weighting the input samples in sequence. By lowering the bias and variance of the individual decision trees in this scenario, boosting enhanced the model's performance and produced a more precise and reliable model.

It is crucial to keep in mind that the poor accuracy of the random forest model could be caused by the model's hyperparameters, including the quantity of decision trees utilised and the depth of each tree. The performance of the model may be enhanced by adjusting these hyperparameters, although this takes substantial testing and optimisation.

Overall, the poor starting accuracy of the random forest model emphasises the significance of experimenting with various machine learning models and methodologies to discover the best match for the particular study field and dataset. It was discovered that enhancing the random forest model was a potential strategy for increasing its accuracy for classifying the genres of Indian music.

An accuracy of 85% was attained by the RNN-LSTM model, matching that of the ANN model. The accuracy graph demonstrated a progressive rise in accuracy over time that finally peaked and persisted throughout the testing session. The RNN-LSTM model had a larger loss than both the ANN and CNN models, suggesting that it made more mistakes while being trained, according to the loss graph.

Overall, this study's findings show that the best models for classifying the musical genres of India are the ANN and RNN-LSTM models. The particular study field, the size of the dataset, and the feature extraction techniques used will all influence the model selection, however. A more thorough understanding of the models' performance is provided by the use of accuracy and loss graphs, which may be helpful for further refining the models for Indian music genre categorization. To increase the precision and effectiveness of the models, future research might investigate the usage of additional machine learning models and feature extraction techniques.

CONCLUSION

In conclusion, machine learning models have shown promise in classifying the musical genres of India. The models with the best accuracy were the ANN and RNN-LSTM, then the CNN model. Despite having lesser accuracy at first, the random forest model was able to perform better because to boosting. For both scholars and lovers as well as professionals in the Indian music business, these models might be useful resources.

The capacity to better comprehend customer tastes and behaviour is one of the key commercial consequences of Indian music genre categorization. To boost user engagement and income, accurate categorization models may assist streaming services and music labels in making targeted suggestions to consumers. Streaming services, for instance, may provide personalised playlists and suggestions that are more likely to keep a user interested by precisely categorising a user's preferred genre.

The possibility for enhanced music creation and promotion has other financial implications. Music producers may develop and promote music that is more likely to connect with their target audience by researching which genres are most popular and which elements make those genres unique. The creation of precise and effective categorization models may also aid in preserving and promoting the rich cultural history of Indian music. Traditional music can be better categorised and catalogued with the use of machine learning algorithms, making it more accessible and discoverable to larger audiences. Traditional Indian music may therefore become more popular and in demand.

Overall, the categorization of Indian music according to genre has important commercial ramifications for the music business, including more focused suggestions, better music creation and marketing, and more traditional music preservation and promotion. The business and cultural heritage alike may benefit even more from more study and development in this field. Indian music genre categorization might have various more ramifications in addition to the commercial ones just mentioned. Its possible use in music instruction is one such consequence. Music instructors may more effectively teach music by using classification models to identify and classify the many genres of Indian music. This may promote a broader appreciation for the wide variety of Indian musical styles and help preserve traditional music.

There is also a chance for cross-cultural interaction. Accurate categorization methods may make it easier for worldwide listeners to find and enjoy Indian music. This may encourage cross-cultural dialogue and mutual respect as well as further the globalisation of Indian music. The creation of categorization models may also aid in addressing the problem of music piracy. The many genres of Indian music may be correctly identified and categorised, allowing for more efficient monitoring and prevention of copyright infringement. The artists and songwriters whose work is being pirated may benefit more from this, and it may also aid in promoting the listening of legal music.

The categorization of Indian music's genres has a variety of effects on both business and society, to sum up. Accurate categorization models may help with music education, intercultural communication, and music piracy avoidance. The creation of such models may also build a broader understanding of the variety of Indian musical genres and aid in preserving and promoting the rich cultural history of Indian music. Further study and development in this field might open up even more opportunities for the Indian music business and society at large as technology and data continue to grow.

BIBLIOGRAPHY / REFERENCES

1. Tzanetakis, G. and Cook, P. (2002) "Musical genre classification of Audio Signals," *IEEE Transactions on Speech and Audio Processing*, 10(5), pp. 293–302. Available at: <https://doi.org/10.1109/tsa.2002.800560>
2. Sharma, H., & Bali, R. S. (2015). Comparison of ML classifiers for Raga recognition. *International Journal of Scientific and Research Publications*, 5(10).
3. Jothilakshmi, S., & Kathiresan, N. (2012). Automatic Music Genre Classification for Indian Music. *International Conference on Software and Computer Applications (ICSCA 2012)*, 41.
4. Guo, G., Wang, H., Bell, D., Bi, Y., & Greer, K. (2003). KNN Model-based Approach in Classification. *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, 986–996. https://doi.org/10.1007/978-3-540-39964-3_62
5. Pachet, F., & Cazaly, D. (2000). A classification of musical genre. In *Proc. RIAO Content-Based Multimedia Information Access Conf.*, Paris, France.
6. Sridhar, R., & Geetha, T. V. (2013). Raga identification of Carnatic music based on the construction of Raga Model. *International Journal of Signal and Imaging Systems Engineering*, 6(3), 172. <https://doi.org/10.1504/ijssie.2013.054795>
7. Sharma, H., & Bali, R. S. (2014). Raga identification of Hindustani music using soft computing techniques. *2014 Recent Advances in Engineering and Computational Sciences (RAECS)*. <https://doi.org/10.1109/raecs.2014.6799544>
8. Kini, S., Gulati, S., & Rao, P. (2011). Automatic genre classification of North Indian Devotional Music. *2011 National Conference on Communications (NCC)*. <https://doi.org/10.1109/ncc.2011.5734697>
9. Choi, K., Fazekas, G., Sandler, M., & Cho, K. (2017). Convolutional recurrent neural networks for music classification. *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. <https://doi.org/10.1109/icassp.2017.7952585>
10. Srivastava, K. (2022). Deep learning techniques for music genre classification and feature importance. <https://doi.org/10.36227/techrxiv.21265965>
11. Manikandan, K., & Mathivanan, G. (2023). An intelligent music genre classification method with feature extraction based on Deep Learning Techniques. *2023 International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT)*. <https://doi.org/10.1109/idciot56793.2023.10053460>