Music Genre Classification Using Multi-Layer Perceptron

Ashish Bhattarai
(904214538)
azb0224@auburn.edu

Nihita Reddy S
(904280752)
nrs0041@auburn.edu

Sudhanshu Kumar
(904284512)
szk0237@auburn.edu

ABSTRACT

Categorizing music files according to their genre is a challenging task in the area of music information retrieval (MIR). A method for categorizing songs or other audio music into different genres is known as the music genre classification model. A more effective and precise model for this classification must be created because people's lives are growing more and more dependent on music, technology, and the internet, all of which are becoming more and more affordable to end users. Our project's goal is to develop a machine learning algorithm that can more accurately detect a song's genre than the existing models. The GTZAN dataset was used to train various categorization models that we constructed for this project. In this project, we compare the performance of three classes of models namely Support Vector Machine classifier, Random Forest (RF) classifier, and two and three layered Multi-Layered Perceptrons (MLP) built from scratch. We train these machine learning classifiers for 3 second and 30 second datasets and compare their performance. The features that contribute the most towards this multiclass classification task are identified. The experiments are conducted on the Audio data set, and we report an accuracy value of 88.7% for 3 layered MLP classifier. All the models showed good accuracy. Our model built from scratch performed better than inbuild models for 3-second dataset however, for 30 sec data, RF performed better.

Keywords: MLP classifier, Neural network, Music genre classification

1. Introduction

People find it harder and harder to manage the songs they listen to as internet music databases and simple access to music information proliferate. The genre, which is determined by elements of the music such rhythmic structure, harmonic content, and instrumentation, is one method for classifying and organizing songs (Tzanetakis and Cook, 2002). For audio streaming services like Spotify and iTunes, being able to automatically categorize and assign tags to the music that is currently in a user's collection depending on genre would be advantageous as individuals demonstrate different preferences in music, yet little is known about the underlying preferences of music.

The demand for precise meta-data for database management and search/storage purposes increases along with the daily volume of music being released. Due to its numerous real-world applications, this discipline has experienced considerable growth during the last ten years. Because of the way that music is arranged, it may be able to improve some cognitive networks by boosting productivity, alertness, and concentration. In the past, music has also been utilized for ritual and religion, social connection, comfort, motivating or organizing physical labor, preserving and transmitting oral information, and expressing mental or physical fitness.

A background in psychology, academic music study, signal processing, informatics, machine learning, or computational intelligence may be required to use or research MIR. Although it's one of the most popular ways to maintain digital music libraries, classifying music is a necessary and extremely challenging undertaking. Music genres are hard to categorize because of their vague and subjective character. The MIR community has been debating this concept for years. Consider the fact that the term "genre" for a certain piece of music was invented later. Every few years, genres are also redefined. They might overlap or develop hierarchies without necessarily being subsets of one another. The sound of music frequently doesn't match genres either. You'll discover that they reflect the time, region, or culture of the musician who made them more.

In our project, we use machine learning models to recognize and categorize music genres by extracting features from the music dataset and using those features to classify songs. Feature extraction involves reducing the number of resources used to describe a large set of data. The key to constructing an effective model is properly optimized feature extraction. Finally, we evaluate our model performances with some inbuilt packages and compare the accuracy and other model evaluation parameters.

1.1. Literature review

Since the early days of the Internet, categorizing music by genre has been a subject of intensive investigation. With the help of supervised machine learning techniques like the Gaussian Mixture model and closest neighbor classifiers, Tzanetakis and Cook (2002) addressed this issue. For this objective, they introduced three sets of features classed as pitch content, rhythmic content, and timbral structure. Music genre classification has also been examined using Hidden Markov Models (HMMs), which have been widely employed for speech recognition applications (Scaringella and Zoia, 2005; Soltau et al., 1998). In Mandel and Ellis (2005), support vector machines (SVMs) using various distance metrics are investigated and compared for genre classification.

In Lidy and Rauber (2005), the authors discuss the contribution of psycho-acoustic features for recognizing music genre, especially the importance of STFT taken on the Bark Scale (Zwicker and Fastl, 1999). Mel-frequency cepstral coefficients (MFCCs), spectral contrast and spectral roll-off were some of the features used by (Tzanetakis and Cook, 2002). A combination of visual and acoustic features are used to train SVM and AdaBoost classifiers in Nanni et al. (2016).

Numerous studies use deep neural network approaches to analyze speech and other types of audio data as a result of the recent success of these systems (AbdelHamid et al., 2014; Gemmeke et al., 2017). Due of the high sampling rate of audio signals, it is difficult to represent audio in the time domain for neural network input. For audio generation jobs, it has been handled by Van Den Oord et al. (2016). The spectrogram of a signal, which records both time and frequency information, is a typical alternative representation. You can think of spectrograms as images and train convolutional neural networks (CNNs) using them (Wyse, 2017). A CNN was created to predict the musical genre using the raw MFCC matrix as input (2010). In Lidy and Schindler (2016), the same goal was accomplished using a constant Q-transform (CQT) spectrogram as the CNN's input.

This study compares two types of machine learning classifiers namely SVM and RF with our baseline model build from scratch. And finally compare the accuracy of these models. This report is organized as follows. Section 2 describes the datasets and methodology used for the task of music genre classification. Also, the proposed models and the implementation details are discussed in this section. The results are reported and discussed in Section 3, followed by the conclusions, challenges and future scopes from this study in Section 4.

Problem statement

One of the primary categories used to categorize millions of music is genre. The tracks are divided into a few different genres. With the most effective methods and algorithms currently available, a futuristic model that improves song classification in the music industry must be created for the current and upcoming generations. to create a machine learning-based model that addresses the problem of automatically classifying music into its relevant genres. to improve the resulting model's accuracy rate in comparison to earlier attempts and current models. It is necessary to create a relevant model for the entertainment sector given the rapidly expanding machine learning capabilities. To lessen or completely replace the manual work involved in determining the genre of music. to create a model that is more accurate to use in real-time applications.

2. Datasets and methodology

The GTZAN Dataset, a popular music genre classification dataset, was the one we chose. One hundred audio snippets, each lasting 30 seconds, are included in a collection of music that includes information on ten distinct musical genres. In addition, it contains 9990 audio samples lasting three seconds. Additionally, each audio file has a visual representation for future image classification. Two CSV files exist that contain information about the audio files. Overall, we discovered that this dataset is excellent for use. The dataset's only issue is that the 30-second clips had fewer samples than its 3-second equivalent, which resulted in poorer accuracy. Each track is in .wav format. It contains audio files of the following 10 genres:

0	Blues	5	Jazz
1	Classical	6	Metal
2	Country	7	Pop
3	Disco	8	Reggage
4	Hip hop	9	Rock

Different genres have different historical and cultural backdrops. Some genres contain a significantly larger variety of musical styles than others. Blues, rock, and metal are examples of certain genres that are closely related. The blues gave rise to rock, which in turn gave rise to metal. The classical genre, which includes considerably older music than the other genres, can be highly distinct in contrast.

2.1. Resources

- The language used is python.

- Libraries

· NumPy: The Python package NumPy is used to manipulate arrays. It enhances Python with strong data structures that provide effective computations with arrays and matrices, and it offers a sizable library of advanced mathematical functions that work with these arrays and matrices. Additionally, it has matrices, Fourier transform, and functions for working in linear algebra. Numerical Python is referred to as NumPy.

· Pandas: Pandas are an open-source library designed primarily for working quickly and logically with relational or labeled data. It offers a range of data structures and procedures for working with time series and numerical data. The NumPy library serves as the foundation for this library. Pandas is quick and offers its users exceptional performance & productivity. The datasets will be loaded from the already-existing storage, which can be an Excel file, CSV file, or SQL database. A Pandas DataFrame can be produced from lists, dictionaries, and lists of dictionaries, among other sources. Python's Pandas library is most frequently used for applications involving data science, data analysis, and machine learning.

· Matplotlib.pyplot: A group of functions known as matplotlib.pyplot enable matplotlib to function similarly to MATLAB. Each pyplot function modifies a figure in some way, such as by creating a figure, a plotting region within a figure, some lines within a plotting area, labeling the plot, etc. The plotting functions are directed to the current axis while other states, such as the current figure and plotting area, are retained in matplotlib.pyplot throughout function calls.

· Scikit-Learn: The most practical Python library for machine learning is scikit-learn. Numerous effective methods for machine learning and statistical modeling, such as classification, regression, and clustering are included in the Sklearn package. Using sklearn, there are several ways to evaluate the precision of supervised models on unobserved data. Scikit-learn is used to extract features from text and images. To assess the precision of a classification, sklearn metrics can also be utilized to compute confusion matrices.

2.2. Machine Learning models

Various ML techniques and recent deep learning techniques are being developed to better model processes in different domains, with varying levels of robustness and inherent limitations. In this project, we have used Support Vector Regression (SVR), Random Forest (RF), and Multi-layer perceptrons (2-layer and 3 layer), and a brief description of each algorithm is as follows:

2.2.1. Support Vector Machine (SVM): Support vector machine (SVM) for regression problems is a well-known supervised learning technique proposed by Vapnik (1999). It is based on structural risk minimization and the statistical learning theory. The fundamental hypothesis of SVR is the nonlinear mapping of the primary data into a higher-dimensional space, so classification becomes more straightforward in the feature space. This approach uses support vectors as a selection criterion, generating the best boundaries to categorize the data (Sujay and Paresh 2014). The kernel is the function to perform linear regression in the feature space (Kalra et al. 2013; Yu P-S et al. 2017). Although SVM can use various kernel functions such as polynomial, linear and sigmoid in SVR, the radial basis function (RBF) performs better than other kernels (Barzegar et al. 2017; Wu and Wang 2009; Yu and Liong 2007). Therefore, we have used the RBF model in the present work. The SVR model with an RBF kernel has epsilon (ε), cost (C), and gamma (γ) as model parameters. ε - loss function to describe the regression vector, C - capacity control parameter,aγ - minimize the model space and monitor the complexity of the output. The SVM model minimizes the expected error of the learning model and decreases the overfitting problem (Yu et al. 2008). The significant limitations of modeling with SVM are commonly related to its difficulty in capturing vital parameters (Choubin et al. 2019).

2.2.2. Random Forest (RF): The RF is the aggregate of decision trees, where each tree is produced from bootstrap training samples (Breiman, 2001). The RF model randomly adopts a bagging approach in identifying features. Each node is randomly separated by choosing the most dominant possible predictors to improve the model accuracy without causing overfitting (Breiman, 2001). The RF method is a robust nonparametric approach for modeling and classifying large nonlinear, noisy, and multivariate correlated data (Mohr et al. 2017; Strobl et al. 2008). A single decision tree has high variance and is prone to noise while showing statistical instability (Zhu and Pierskalla, 2016). Statistical instability means that a small perturbation in the training data leads to substantial changes in model outputs. Bootstrap aggregation is used to eliminate over-fitting, and statistical instability, which builds many decision trees by randomly sampling the observed dataset with replacement (Mohr et al. 2017). Predictions for unobserved data can be obtained by averaging forecasts from an ensemble of trees or forests. The RF method advances bootstrap aggregation by using a subset of training data and randomly sampled input variables or features to split tree nodes (Zhu and Pierskalla, 2016).

2.2.3. Multi-layer perceptron (MLP): MLPs are artificial neural networks that feed forward. A group of algorithms known as neural networks is modeled after the human brain. They produce a label after recognizing patterns in numerical input data. Before neural networks can understand different inputs, such as images, audio, and text, these inputs must first be converted into numerical data. It is a neural network with a non-linear input-to-output mapping.

An input, output, and one or more hidden layers—each with several neurons layered together—make up a multilayer perceptron. The neurons in a Multilayer Perceptron can employ any arbitrary activation function, (Mastorakis 2009) in contrast to neurons in a Perceptron, which must have an activation function that enforces a threshold, such as ReLU or sigmoid.

Given that inputs and initial weights are mixed in a weighted sum and are both subject to the activation function, Multilayer Perceptron falls within the category of feedforward algorithms. (Carolina Bento 2021) However, the distinction is that each linear combination is carried over to the following layer. The output of each layer's computation and internal representation of the data is fed to the layer below it. This passes through all hidden layers and ends at the output layer. But it goes beyond that. The method would not be able to discover the weights that minimize the cost function if it merely computed the weighted sums in each neuron, transmitted findings to the output layer, and stopped there. No real learning would occur if the algorithm simply computed one iteration. Backpropagation is useful in this situation.

The Multilayer Perceptron's learning technique, backpropagation, (D. Rumelhart 1986) enables it to incrementally change the network's weights with the aim of minimizing the cost function. For backpropagation to function successfully, there is one strict need. It is necessary for both the threshold function, such as ReLU and the function that mixes inputs and weights in a neuron to be differentiable. Since Gradient Descent is frequently employed as an optimization function in MultiLayer Perceptron, these functions must have a bounded derivative. The gradient of the Mean Squared Error is calculated across all input and output pairs in each iteration after the weighted sums have been passed through all layers. The gradient's value is then updated in the first hidden layer's weights to propagate it back. The weights are transmitted back to the neural network's origin. The procedure continues until the gradient for each input-output pair converges, when the freshly computed gradient does not differ from the preceding iteration by more than a predetermined convergence threshold.

Activation functions (σ):

The activation used in this project are ReLU, sigmoid, and softmax function. Mathematically, they can be expressed as follows:

ReLU:

$$\sigma(x) = \begin{cases} 0 & \text{for } x \le 0 \\ x & \text{for } x > 0 \end{cases} = \max(0, x)$$

Sigmoid:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Softmax:

$$\sigma(x) = \frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}} \quad \text{for } i = 1, \dots, J$$

Feed forward process can be mathematically expressed as:

$$z = W^T \cdot X = \sum_i x_n w_{in}$$

$$a = \sigma(z)$$

Where, W , and X are weightage and features respectively. The term a is the output vector from the hidden layer.

Cost function is given as:

$$E = \frac{1}{2} \sum_k (a_k - y_k)^2$$

Where, y_k and E are the target labels and error, respectively.

The weightage (W) can be updated using backpropagation method which is given by:

$$\mathrm{W_{t+1}} = \mathrm{W_t} - \frac{\partial \mathrm{E(W)}}{\partial \mathrm{W_t}}$$

2.3. Hyperparameters Tuning

The hyperparameters were tuned for both two-layer and three-layer MLP classifiers for music genre classification. Initially, for training and testing the data, the original data, i.e., 3 secs and 30 secs data files were split into 80% for training and 20% for testing.

The hyperparameters for MLP classifiers are a number of hidden layer nodes, learning rate, number of epochs, and combination of activation functions, etc. Both, two-layer and three layer-models were implemented for training the MLP classifier. Initially, Different combinations of activation functions were tried for the two-layer and three-layer MLP classifier. The accuracy was found to be highest for ReLU-softmax, and ReLU-ReLU-softmax combinations for both two-layer and three-layer MLP classifiers respectively.

After that, the learning rate versus accuracy was plotted. The learning rate that gave the maximum accuracy was stored. The number of hidden layer nodes for the two-layer model was varied from 10 to 120 with the increment of 10 and plotted against the optimal accuracy and corresponding learning rate. The number of hidden layer nodes and the learning rate with the highest optimal accuracy were selected.

The same experiment was repeated for a different number of epochs from 100 to 300 with an increment of 50. Finally, the number of epochs was selected in such a way that gave the maximum highest optimal accuracy.

The same process was repeated to tune the three-layer MLP classifier. The number of hidden layers (except the output layer) nodes were selected to be the same.

2.4 Software Description

- For MLP Scratch Model

1) Read Data – Initially, reading of the two csv files of GTZAN dataset (30 second dataset and 3 second dataset), the unnecessary columns are dropped ('filename') and convert the files into NumPy arrays and scaling of the data is done.

   Note: Firstly, we have done the for 30second dataset (after finding the accuracies and optimal learning rate, confusion matrix) then we applied the same procedure for the 3second dataset as well.

2) Split the data – then the data is divided into different sets i.e., training set, validation set and testing set. To avoid overfitting.

3) One-hot Encoding - One-hot encoding is a technique for converting a set of categorical characteristics into numerical dummy features. We have defined a function which converts the list of numbers into a matrix of one-hot encoded vectors.

4) Finding optimal learning rate, epochs, and test accuracy for two-layer model

   The program implements a two- layer neural network function, it starts with initializing the parameters.

- Now, we perform the feed-forward operation (linear activation forward) in which it starts with utilizing a parameterized input, weight, and bias (starting from w1, b1). The result of the linear transformation of the input by the weight matrix, with the bias vector added, is returned (forward function does that) and based on which activation function we are using, we compute that function. We have used SoftMax activation function and ReLu activation functions.

- Generally, the activation function is used to introduce the non-linearity to a neuron's output. The neural network's unprocessed outputs are converted into a vector of probabilities—basically, a probability distribution over the input classes, this is what a SoftMax activation function does and as per the ReLu activation function, any negative input causes the function to return 0, but any positive value x causes it to return that value. As a result, an output with a range of 0 to infinite is produced. In this part (feed-forward), we calculate from input layer to output layer and calculate the cost.

- For weight 1, bias 1, the ReLu activation function is being used and for weight 2 and bias 2 is done by SoftMax activation function.

- Then we backpropagate which are frequently employed to train feedforward neural networks. The gradient of the loss function with respect to the network weights is easily computed. The linear activation backward functions take the cost function's derivative about a layer's input and returns the derivative with respect to the layer's weights and biases. The gradients of the weights and bias are found and then updating of parameters is done.

- For each node we are calculating the accuracy for a particular learning rate. Hence, the optimal learning rate is defined from that. Test accuracy is also derived in this part of program. (Two-layer scratch model).

5) Plotting a graph for number of hidden layer nodes vs Accuracy with no of epochs being 200 and plotting a graph for no of hidden layer nodes vs learning rate for epochs being 300.

6) Then with respect to optimal no of epochs and no of hidden nodes, plotted a graph for learning rate and accuracy. We then constructed the confusion matrix for the true label and predicted label.

We have done similar process for finding the test accuracy, optimal learning rate, no of hidden nodes and no of epochs and plotting the graphs number of hidden layer nodes vs Accuracy with no of epochs being fixed and for plotting a graph for no of hidden layer nodes vs learning rate and learning rate vs accuracy and confusion matrix with respect to three-layer model.

And this whole program is repeated for 3second dataset as well (with the use of two-layer and three-layer model to find accuracy and confusion matrix).

- Baseline Model

For MLP classifier, since in the same file we are doing the baseline model, so basic step of reading csv file is done. Then we used sklearn MLP classifier function and predicted the accuracy and confusion matrix for both two layer and three-layer model. This is again done for the 30second dataset and 3second dataset.

For SVM classifier: we imported the three and thirty second dataset. Then we scaled the data using standard scalar setting the mean to zero and standard deviation to one. Then we spit the datasets for testing and training sets. Finally, we computed the accuracy score and confusion matrix.

For RF classifier: we imported the three and thirty second dataset. Then we scaled the data using min-max scalar setting the mean to zero and standard deviation to one. Then we spit the datasets for testing and training sets. Finally, we computed the accuracy score and confusion matrix.

- **Software Challenges:**

For the MLP implementation, initially, the data was used without one hot encoding. Doing so caused low accuracy for the MLP classifier. Thus, the data labels were one hot encoded to binary variables for better performance of MLP classifier.

Attribute values of our dataset were in a different scale. As a result, they do not contribute equally to training MLP classifier. Thus, it was necessary to standardize the data using standardscalar() function in python. It standardizes each attribute data individually and sets the mean to zero and standard deviation to 1.

2.5. Evaluation matrics:

	Predicted: NO	Predicted: YES
Actual: NO	True Negative	False Positive
Actual: YES	False Negative	False Negative

$$Accuracy = \frac{True\ Positive + True\ Negative}{Total}$$

The model's performance across all classes is often described by its accuracy metric. When every class is equally important, it is useful. It is determined by dividing the total number of guesses by the number of predictions that were correct.

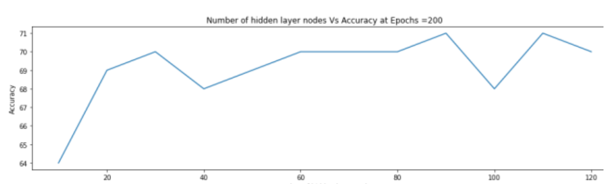
In a N x N matrix, where N is the number of target classes, a confusion matrix is used to assess how well a classification model is working. The machine learning model's predictions are put up against the actual target values in a matrix.

3. Results and Discussion

3.1. MLP classifier (Scratch)

The tuned hyperparameters were selected for both 30 secs and 3 secs datasets and results such as training accuracy and testing accuracy were compared for both two-layer and three-layer MLP classifiers. Figure 1A shows the variation of optimal accuracy with several hidden layer nodes for 30 secs of data with a two-layer MLP classifier. While Figure 1B, shows the variation of learning rate with optimal accuracy at a tune number of epochs, and hidden layer nodes. The maximum testing and training accuracy obtained for 30 secs data using two-layer MLP was 71% and 100%, respectively.

(A)



(B)

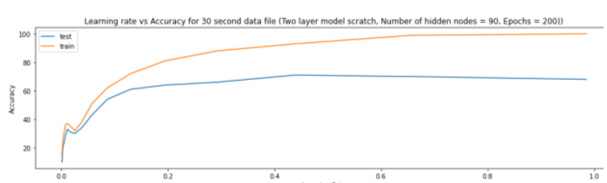
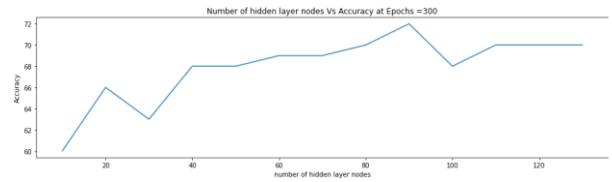


Figure 1: (A): Hidden layer nodes versus optimal accuracy (30 secs data, two-layer MLP classifier, number of hidden layers 90, epochs 200). (B): Learning rate versus accuracy.

(A)



(B)

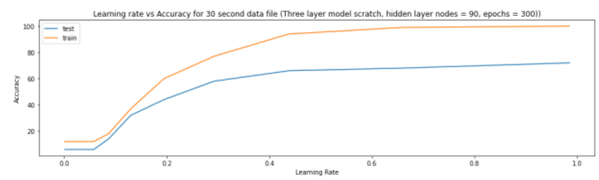
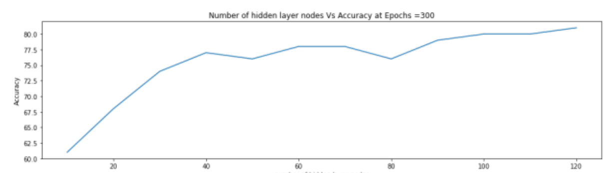


Figure 2: (A): Hidden layer nodes versus optimal accuracy (30 secs data, three-layer MLP classifier, number of hidden layers 90, epochs 300). (B): Learning rate versus accuracy.

(A)



(B)

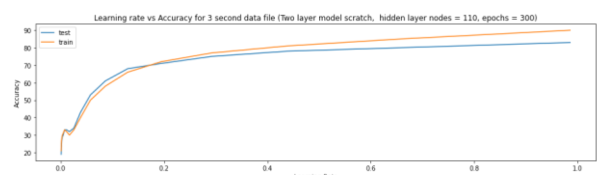
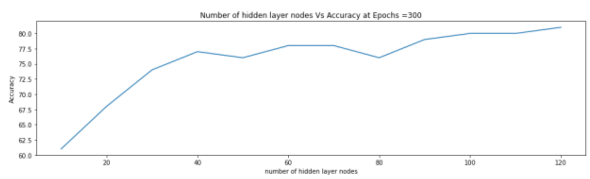


Figure 3: (A): Hidden layer nodes versus optimal accuracy (3 secs data, two-layer MLP classifier, number of hidden layers 110, epochs 300). (B): Learning rate versus accuracy.

A similar plot was shown for 30 secs data using the three-layer model in Figure 2. The maximum testing and training accuracy obtained for 30 secs data using three-layer MLP was 72% and 100%, respectively. The increase in testing accuracy when using a two-layer to three-layer model is 1%. The computation cost for training three-layer MLP is higher than two-layer MLP. Thus, it might not be efficient enough to use three-layer MLP in training the data.

Figure 5 and 6 shows the confusion matrix for 30 secs data for two-layer and three-layer MLP classifier, respectively. Similarly, Figure 7 and 8 shows the confusion matrix for 3 secs data for two-layer and three-layer MLP classifier, respectively.

(A)



(B)

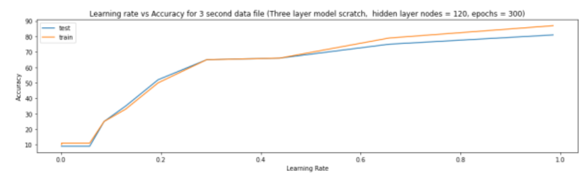


Figure 4: (A): Hidden layer nodes versus optimal accuracy (3 secs data, three-layer MLP classifier, number of hidden layers 120, epochs 300). (B): Learning rate versus accuracy.

Table 1: Hyperparameters for 30 secs data

Parameter for 30 secs data	**Two-layer MLP**	**Three-layer MLP**
Learning rate	0.43789	0.98526
Hidden layer nodes	90	90 – 90
Activation function combination	ReLU - Softmax	ReLU – ReLU - Softmax
Epochs	200	300
Test accuracy	71%	72%
Training accuracy	100%	100%

Table 2: Hyperparameters for 3 secs data

Parameter for 3 secs data	**Two-layer MLP**	**Three-layer MLP**
Learning rate	0.017085	0.98526
Hidden layer nodes	110	120-120
Activation function combination	ReLU – Softmax	ReLU – ReLU – Softmax
Epochs	300	300
Test accuracy	82%	81%
Training accuracy	90%	90%

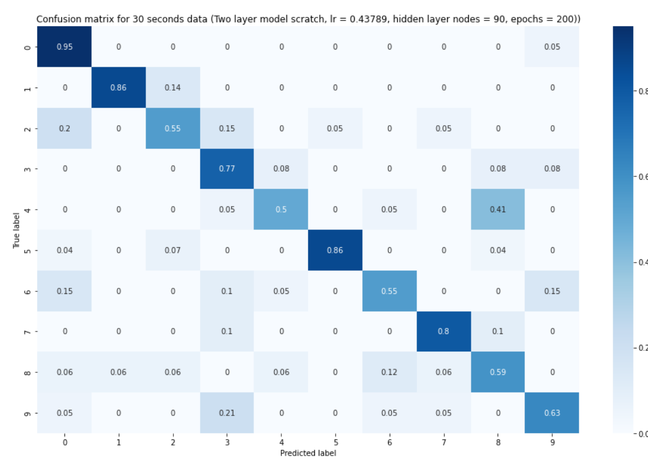


Figure 5: Confusion matrix for 30 secs data with two-layer MLP written from scratch.

Similarly, Figure 3 and Figure 4, demonstrate the results for 3 secs data when using two-layer, and three-layer MLP, respectively. The maximum testing and training accuracies obtained for two-layer MLP are 82% and 90% respectively. Also, the maximum testing and training accuracies obtained for three-layer MLP are 81% and 90% respectively. Thus, it can be seen that two-layer MLP works best for 3 secs dataset. The summary of tuned hyperparameters and algorithm accuracies are summarized in Table 1 and Table 2.

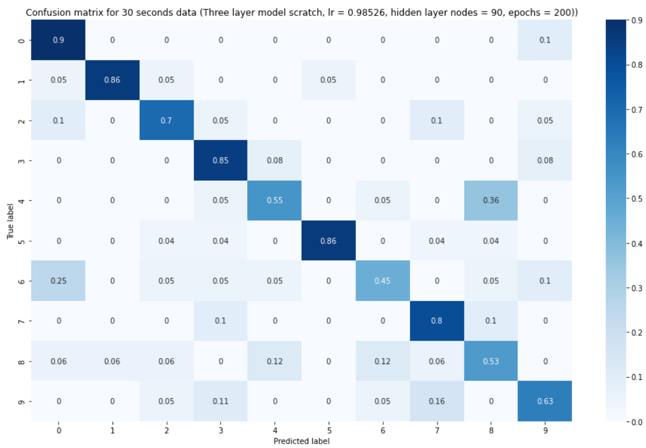


Figure 6: Confusion matrix for 30 secs data with three-layer MLP written from scratch.

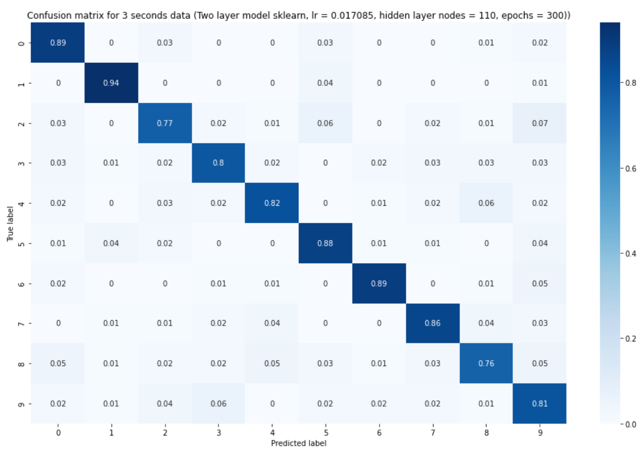


Figure 7: Confusion matrix for 3 secs data with two-layer MLP written from scratch.

Figure 8: Confusion matrix for 3 secs data with three-layer MLP written from scratch.

3.2. MLP classifier (SKlearn)

In order to validate the implementation of two-layer and three-layer MLP (developed from scratch) in music genre classification, the two-layer and three-layer MLP from SKlearn library was used to classify the same data at same hidden layer nodes and number of epochs as for MLP developed from scratch. Figure 9 and 10 shows the learning rate versus accuracy for 30 secs dataset for two-layer and three-layer MLP (from SKlearn), respectively. The maximum testing accuracy obtained for 30 secs data with two-layer and three-layer MLP are 74.5% and 71.5%, respectively.

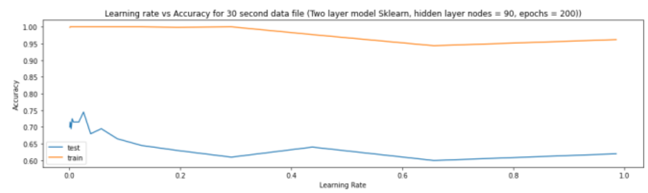


Figure 9: Learning rate versus accuracy for two-layer MLP classifier (SKlearn) 30 secs data.

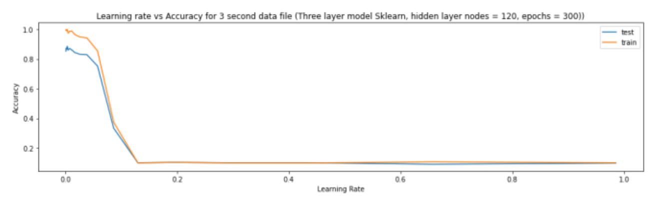


Figure 10: Learning rate versus accuracy for three-layer MLP classifier (SKlearn) 30 secs data.

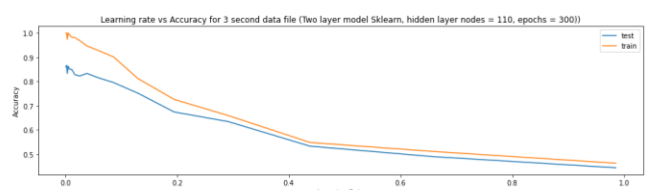


Figure 11: Learning rate versus accuracy for two-layer MLP classifier (SKlearn) 3 secs data.

Similarly, Figure 11 and 12 shows the learning rate versus accuracy for 3 secs dataset for two-layer and three-layer MLP (from SKlearn), respectively. The maximum testing accuracy obtained for 3 secs data with two-layer and three-layer MLP are 86.58% and 88.68%, respectively.

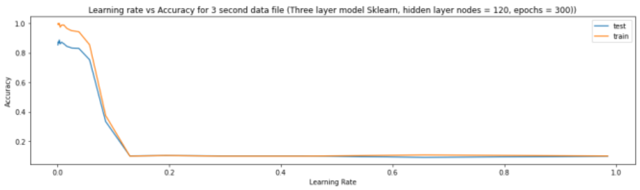


Figure 12: Learning rate versus accuracy for three-layer MLP classifier (SKlearn) 3 secs data.

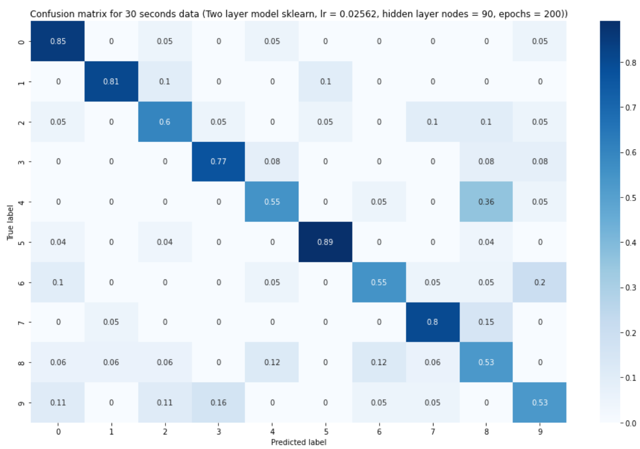


Figure 13: Confusion matrix for 30 secs data with two-layer MLP classifier (SKlearn).

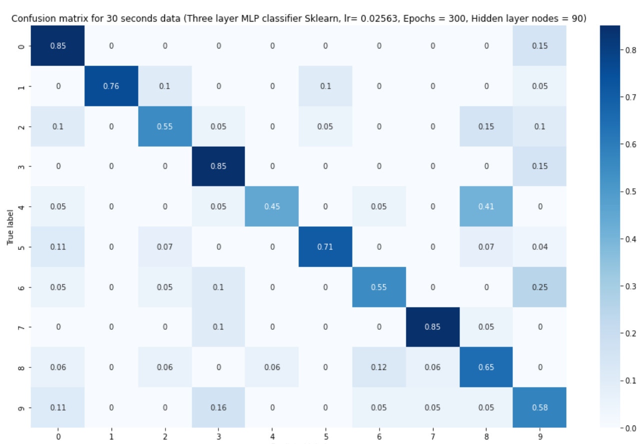


Figure 14: Confusion matrix for 30 secs data with three-layer MLP classifier (SKlearn).

Figure 13 and 14 shows the confusion matrix for 30 secs data for two-layer and three-layer MLP classifier, respectively. Similarly, Figure 15 and 16 shows the confusion matrix for 3 secs data for two-layer and three-layer MLP classifier, respectively

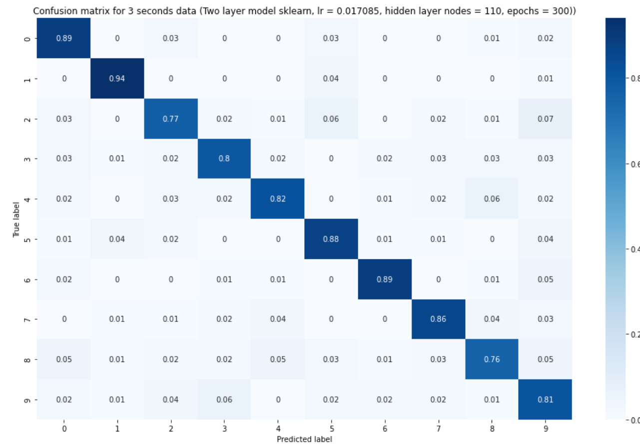


Figure 15: Confusion matrix for 3 secs data with two-layer MLP classifier (SKlearn).

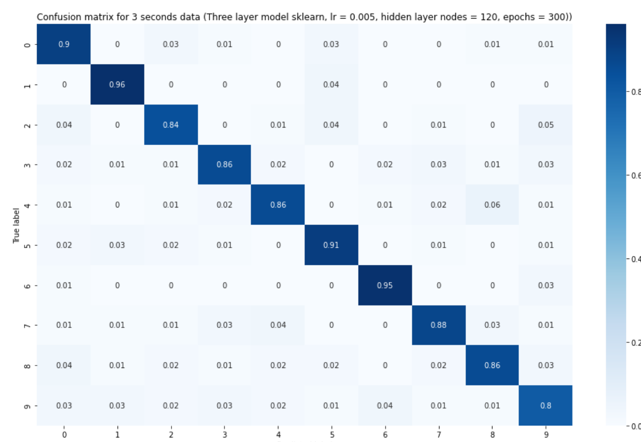


Figure 16: Confusion matrix for 3 secs data with three-layer MLP classifier (SKlearn).

Figure 17 and 18 shows the confusion matrix for 30 secs and 3 sec data for RF classifier, respectively. Similarly, Figure 19 and 20 shows the confusion matrix for 30 sec and 3 secs data for SVM classifier, respectively

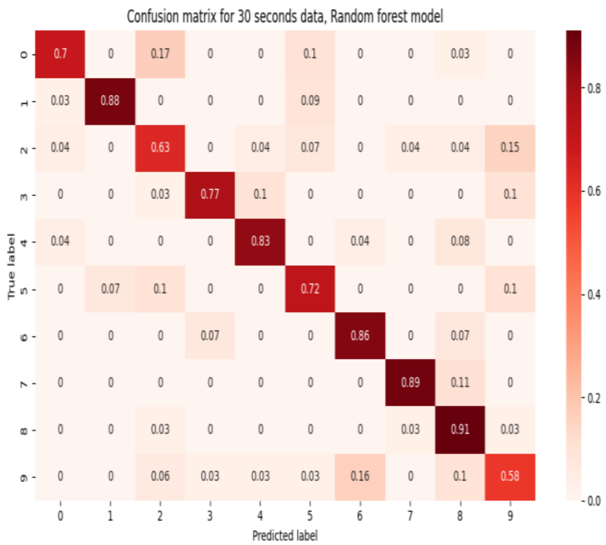


Figure 17: Confusion matrix for 30 secs data with RF model.

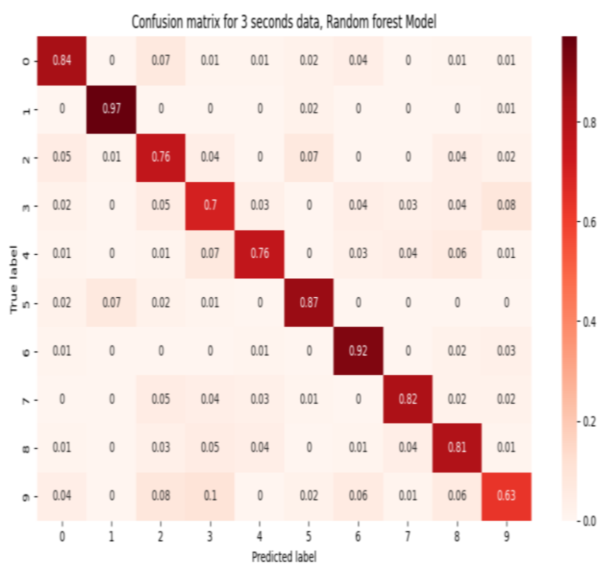


Figure 18: Confusion matrix for 3 secs data with RF model.

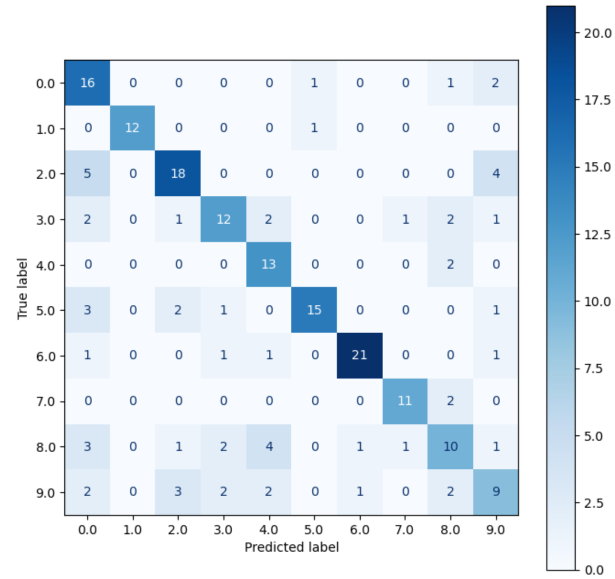


Figure 19: Confusion matrix for 30 secs data with SVM model.

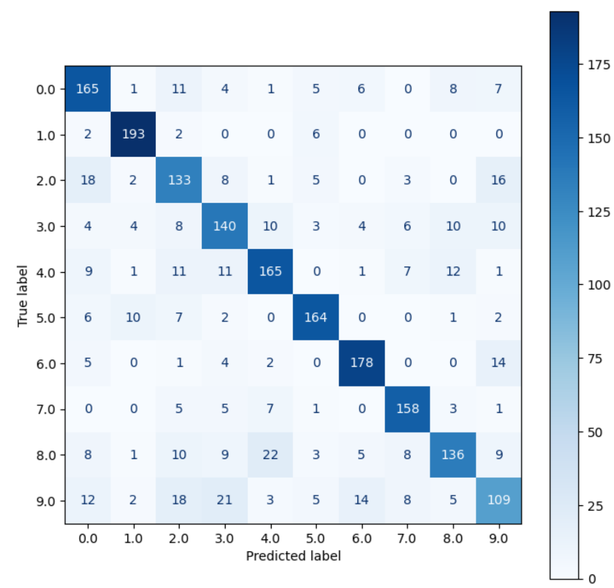


Figure 20: Confusion matrix for 3 secs data with SVM model.

4. Conclusions future scope

The classification of musical genres is an important and extensively researched issue. The strategies for categorizing music genres grow as a result of various advancements. To categorize and contrast the musical genres, we were able to put three models into practice. From start, we developed a RandomForest classifier, an SVM classifier with sklearn, a two-layer MLP, and a three-layer MLP. With respect to the 30 second data characteristics, the RandomForest model and our MLP models both exhibited the best levels of accuracy. Rock was the genre that had been misclassified the most, as we could see. Rock and blues both have elements in common with pop music. As a result, it is more frequently misclassified. We think that the classification will significantly advance with more distinctive traits.

Accuracy	Testing	
	30 Sec data	**3 Sec data**
SVM	68.0%	77.0%
RF	78.0%	80.0%
MLP 2 layer	74.5%	86.6%
MLP 3 layer	71.5%	88.7%

Each approach delivered outstanding outcomes. Our original neural network model came close to matching the sklearn model in terms of success. We think that each of these models offers practical approaches to identifying the musical genres. As a result of the features' greater specificity compared to the 30 second data features, we find that utilizing 3 second data features significantly improve the model. Furthermore, due to its great accuracy, we draw the conclusion that the MLP model is the best model for classifying music genres out of the three. We intend to investigate which characteristics aid in enhancing rock classification in the future.

We had several issues with our analysis that may have prevented us from getting better scores, even though our model provided accurate answers in some cases. The quantity of the 30-second data collection presented us with our first and biggest problem. We only had 10 clips for each genre, however the 100 samples in the 3 second dataset totaled 200 clips. This can result in the 30 second clips being undersized. The accuracy trend in our scratch and sklearn models against the learning rate was the second issue we found. Our scratch model's accuracy rose steadily when the learning rate was raised. On the other hand, with our sklearn model, the accuracy declined in an unexpected way. Despite the fact that both models had good accuracy, this was alarming. The problem can be in the scratch neural network, or it might be related to the different activation functions.

Acknowledgements

We would like to thank all the people who helped us with this project, without their support and guidance it wouldn't have been possible. We appreciate Mr. Donji and Mrs. Souvika for their continuous guidance and supervision which has provided a lot of resources needed in completing our project. Our special thanks to Dr. Santu for all his support.

Author contributions

Ashish Bhattarai: Worked on implementation of two-layer and three-layer MLP classifier (MLP From scratch and from Sklearn both). Executed hyperparameters tuning and result plotting. Worked on writing MLP classifier theory, result and discussion part of the report.

Nihitha Reddy S: Worked on implementation of random forest classifier and two-layer MLP from Sklearn and implementation of two-layer MLP classifier (scratch). Conducted parameter initialization and parameter update with feed-forward and backward functions. In the project report, wrote problem description, software description, and part of the methodology (Dataset, Resources used, and MLP theory) and helped with other parts of report.

Sudhanshu Kumar: Conception, implementation of support vector machine classifier model and helped in coding MLP classifier, generation of some plots. Also, wrote the abstract, introduction, a part of methodology, and conclusions, challenges and future scopes of the manuscript of the project. Finally helped in formatting the final version of manuscript.

References

Almeida LB. Multilayer Perceptrons, Handbook of Neural Computation, IOP Publishing Ltd. and Oxford University Press, 1997.

Barzegar R, Moghaddam AA, Adamowski J, Fijani E (2017) Comparison of machine learning models for predicting fluoride contamination in groundwater. Stoch Env Res Risk A 31(10):2705–2718

Breiman L.J.M.l. (2001) Random Forests 45 (1):5-32, https://doi.org/10.1023/A: 1018054314350

Carolina Bento , Multilayer Perceptron Explained with a Real-Life Example and Python Code: Sentiment Analysis, 2021.

Choubin B, Moradi E, Golshan M, Adamowski J, Sajedi-Hosseini F, Mosavi A (2019) An ensemble prediction of flood susceptibility using multivariate discriminant analysis, classification and regression trees, and support vector machines. Sci. Total Environ 651:2087-2096

D. Rumelhart, G. Hinton, and R. Williams. Learning Representations by Back-propagating Errors. Nature 323 (6088): 533–536 (1986).

David Kosiur. 2001. Understanding Policy-Based Networking (2nd. ed.). Wiley, New York, NY..

Ian Editor (Ed.). 2007. The title of book one (1st. ed.). The name of the series one, Vol. 9. University of Chicago Press, Chicago. DOI:https://doi.org/10.1007/3-540-09237-4.

Kalra A, Ahmad S, Nayak A (2013) Increasing streamflow forecast lead time for snowmelt-driven catchment based on large-scale climate patterns. Adv Water Resour 53:150–162

Marius-Constantin Popescu, Valentina E. Balas, Liliana Perescu-Popescu, Nikos Mastorakis (2009) . Multilayer Perceptron and Neural Networks.

Mohr CH, Manga M, Wang CY, Korup O (2017) Regional changes in streamflow after a megathrust earthquake. Earth Planet. Sci. Lett. 458:418–428

Patricia S. Abril and Robert Plant, 2007. The patent holder's dilemma: Buy, sell, or troll? Commun. ACM 50, 1 (Jan, 2007), 36-44. DOI: https://doi.org/10.1145/1188913.1188915 .

Sten Andler. 1979. Predicate path expressions. In Proceedings of the 6th. ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL '79). ACM Press, New York, NY, 226-236. DOI:https://doi.org/10.1145/567752.567774

Sujay RN, and CD Paresh (2014) Support vector machine applications in the field of hydrology: A review. Appl. Soft Comput. J. 19 (Jun): 372–386

Vapnik V (1999) The nature of statistical learning theory. 2nd ed. Berlin: Springer.

Wu K-P, Wang S-D (2009) Choosing the kernel parameters for support vector machines by the inter-cluster distance in the feature space. Pattern Recogn 42(5):710–717.

Yu L, Wang S, Lai KK (2008) Forecasting crude oil price with an EMD-based neural network ensemble learning. paradigm Energy Econ. 30 (5):2623-2635

Yu P-S, Yang T-C, Chen S-Y, Kuo C-M, Tseng H-W (2017) Comparison of random forests and support vector machine for real-time radar-derived rainfall forecasting. J Hydrol 552:92–104

Yu X, Liong S-Y (2007) Forecasting of hydrologic time series with ridge regression in feature space. J Hydrol 332(3–4):290–302

Zhu J, Pierskalla WP (2016) Applying a weighted random forests method to extract karst sinkholes from LiDAR data. J. Hydrol. 533:343–352