Music Audio Classification Using ML and DL Techniques

॥वसुधैव कुटुम्बकम्॥

THESIS SUBMITTED TO

Symbiosis Institute of Geoinformatics

FOR PARTIAL FULFILMENT OF THE M.Sc. DEGREE

By

SHRISHTI SINGH

(2020-22 / 20070243031)

Symbiosis Institute of Geoinformatics

Symbiosis International (Deemed University)

5th Floor, Atur Centre, Gokhale Cross Road

Model Colony, Pune-411016

CERTIFICATE

Certified that this thesis titled '**Music Audio Classification using ML and DL techniques**' is a Bonafide work done by Miss. Shrishti Ramnarayan Singh, at Symbiosis Institute of Geoinformatics, under our supervision.

Supervisor, Internal

Dr. Rajesh Dhumal

Symbiosis Institute of Geoinformatics

Index

Acknowledgement

I express my sincere gratitude to my internal guide, Dr. Rajesh Dhumal, for his guidance in the completion of my project. I would like to extend my gratitude to the Director of Symbiosis Institute of Geoinformatics Dr T. P. Singh for providing such opportunities of performing projects and executing them. I must thank the teaching staff of M.Sc. Data Science and Spatial Analytics as I was able to accomplish the project on basis of what I learnt from the courses taught by them.

List of Figures

List of Tables

Abbreviations List

1. FT:	Fourier Transformation
2. FFT:	Fast Fourier Transformation
3. STFT:	Short Term Fourier Transformation
4. MFCCs:	Mel-frequency cepstral coefficients
5. ANN:	Artificial Neural Networks
6. CNN:	Convolutional Neural Networks
7. RNN:	Recurrent Neural Networks
8. LSTM:	Long Short Term Memory

Preface

There are various online musical platforms that give access to musical tracks to the users. A large number of musical tracks are added on the daily basis to various platforms. There are various genres under which the musical tracks can fall under. Such recommendation systems to have user-friendly use of such online platforms and applications the system must be able to classify the musical tracks as per their genres or as per their users's choice. Various applications make use of such recommendation systems like Spotify, jio savan, YouTube Music, Goggle play Music and more. My study focuses on how these musical tracks can be classified as per the music genre basis on what kind of audio data is to be classified. My study has focused on both the CSV audio data and the original audio data file classification techniques. Both Machine Learning algorithms and Deep learning techniques have been studied and their performances have been compared. The study focuses on which technique is better to perform music audio classification and what are the vital features of the audio file can be used for classification.

Introduction

It’s the stated fact that the world has been revolutionized by the growing technologies. Today the most popular way to listen to music is through Online streaming. The global Music Streaming market size by the end of 2021 has been $29.5B as studied by GrandViewResearch. Every music streaming platform should be able to correctly classify the music as per their genres or their playlists in order to provide ease in the usage of the platform for their users. In order to classify the music, we require certain features basis on which musical files could get classified. Tempo, pitch, chord, instrument timber, and many of such features allows to recognize the musical style. But its difficult to extract these features. While the spectral features of the audio are easier to work with. Hence the spectro-temporal patterns can be studied and used for classification.

The GTZAN dataset has been widely used for research purposes in order to study the Music Audio files and their features and is openly available. To work upon audio data, it is important to understand the audio features and patterns that can be used to distinguish it from other genres (Tzanetakis & Cook, 2002).In order to automatically classify the music audio files, various machine learning and deep learning techniques can be used. Detecting and grouping audio files of similar kinds of music genres is a crucial part of music recommendation systems and in generating the playlist as per the individual’s interest.

In my study, I have experimented with various Machine learning and Deep learning techniques in order to study the performances of the algorithms with the audio files. The classification of the music audio files as per their genres has been performed and the performances of each algorithm have been studied and compared.

Literature Review

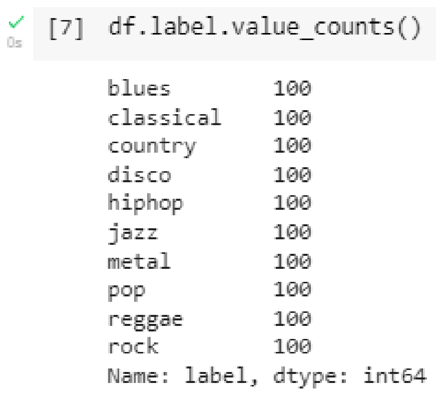
There are various papers which show the study done on the audio data classification. Various attempts have been made using both Machine learning and deep learning approaches to perform audio classification. The original authors of the dataset George Tzanetakis and Perry Cook have tried to find out the features on the basis of which the classification can be performed like spectral features and MFCC's (Ceylan et al., 2021). A recent study done by the authors Gautam Chettiar and Kalaivani S focuses on which feature out of FFT and MFCC is best approach to work with deep learning (Bhatia et al., 2021). In another research study done a Dilated CNN was used . Various research have been done to study and experiment various Machine Learning and Deep Learning methods and what features of the audio data would be helpful to perform classification. (Li et al., 2018)

Dataset and Features

1. Dataset Description

GTZAN is a public dataset widely used for research purposes. The dataset comprises 10 types of genres and each genre folder consists of 100 audio files. In total there are 1000 audio files in the dataset and each audio file is 30 seconds long. The 10 different genres present in the GTZAN are Blues, Classical, Country, Disco, Hiphop, Jazz, Metal, Pop, Reggae, and Rock.

The dataset comprises of three different types of data folders. Genres Orignal which consists of the audio dataset as per the genres, images original which consists of image representation of the audio files as per the genres and a CSV file which consists of the multiple features of the audio files in tabular format.



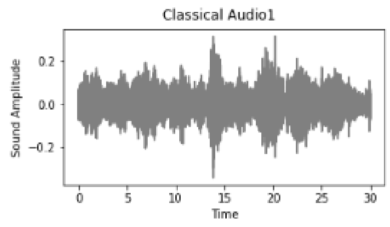
```
[7] df.label.value_counts()

blues        100
classical    100
country      100
disco        100
hiphop       100
jazz         100
metal        100
pop          100
reggae       100
rock         100
Name: label, dtype: int64
```

Fig(1): music genres

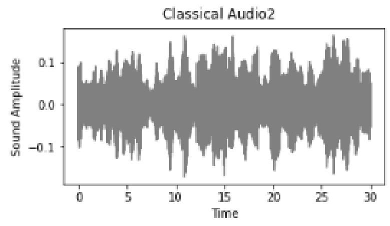
2. Data Features

Audio waves of different genres:

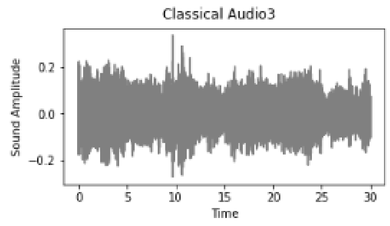
The following figures represent the sound waves of the audio files. Three sample sound waves from each genre is displayed below. It is observed that the audio waves of the audio files from same genre exhibit similar kind of waveforms. Having similar kind of waveforms means they have similar kind of properties and features which can be used to distinguish them from the other genre audio file. For example, the waveform of the genre country is quite stable throughout the time while the genre Jazz shows lot of variations in the waveform.

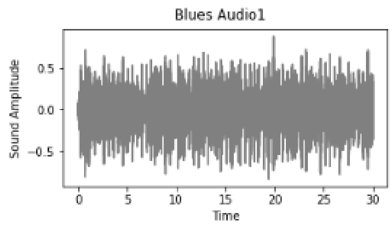

Classical Audio1
Sound Amplitude
Time

```
<Figure size 432x288 with 0 Axes>
```

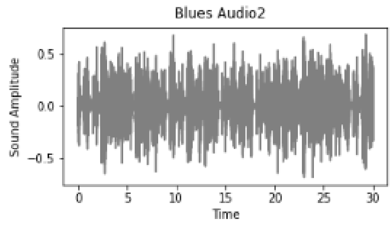

Classical Audio2
Sound Amplitude
Time

```
<Figure size 432x288 with 0 Axes>
```

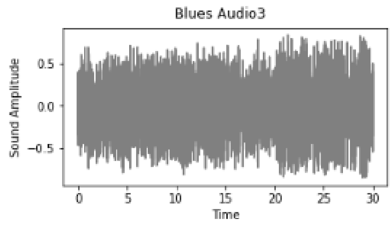

Classical Audio3
Sound Amplitude
Time

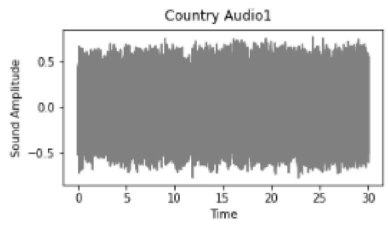

Blues Audio1
Sound Amplitude
Time

```
<Figure size 432x288 with 0 Axes>
```

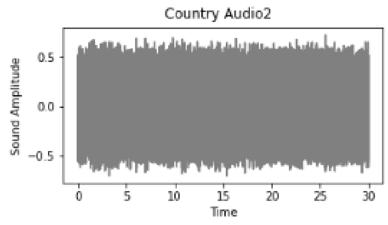

Blues Audio2
Sound Amplitude
Time

```
<Figure size 432x288 with 0 Axes>
```

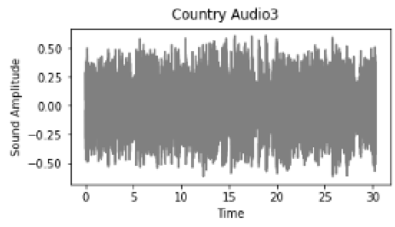

Blues Audio3
Sound Amplitude
Time

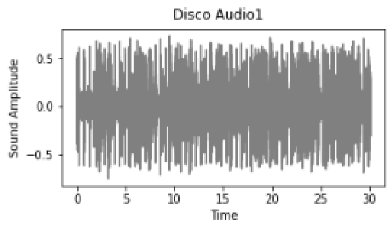

Country Audio1
Sound Amplitude
Time

```
<Figure size 432x288 with 0 Axes>
```

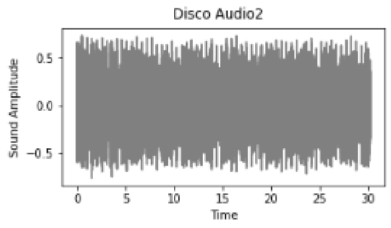

Country Audio2
Sound Amplitude
Time

```
<Figure size 432x288 with 0 Axes>
```

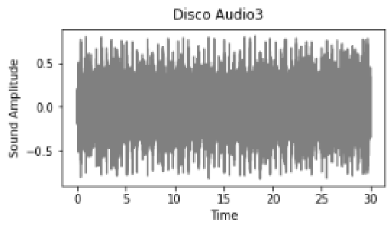

Country Audio3
Sound Amplitude
Time

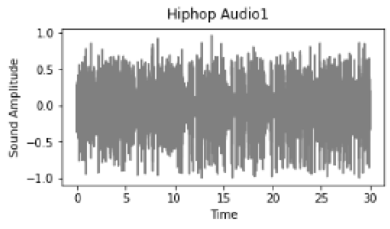

Disco Audio1
Sound Amplitude
Time

```
<Figure size 432x288 with 0 Axes>
```

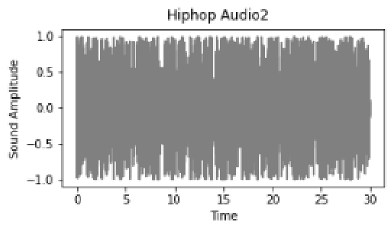

Disco Audio2
Sound Amplitude
Time

```
<Figure size 432x288 with 0 Axes>
```

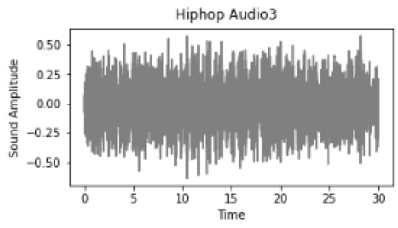

Disco Audio3
Sound Amplitude
Time

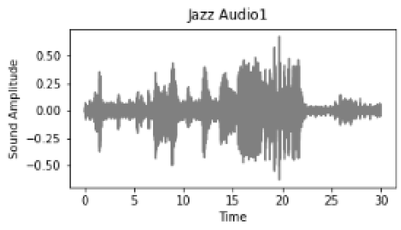

Hiphop Audio1
Sound Amplitude
Time

```
<Figure size 432x288 with 0 Axes>
```

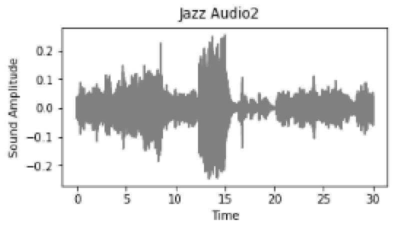

Hiphop Audio2
Sound Amplitude
Time

```
<Figure size 432x288 with 0 Axes>
```

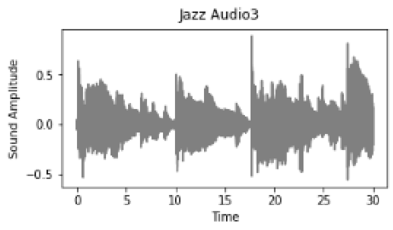

Hiphop Audio3
Sound Amplitude
Time

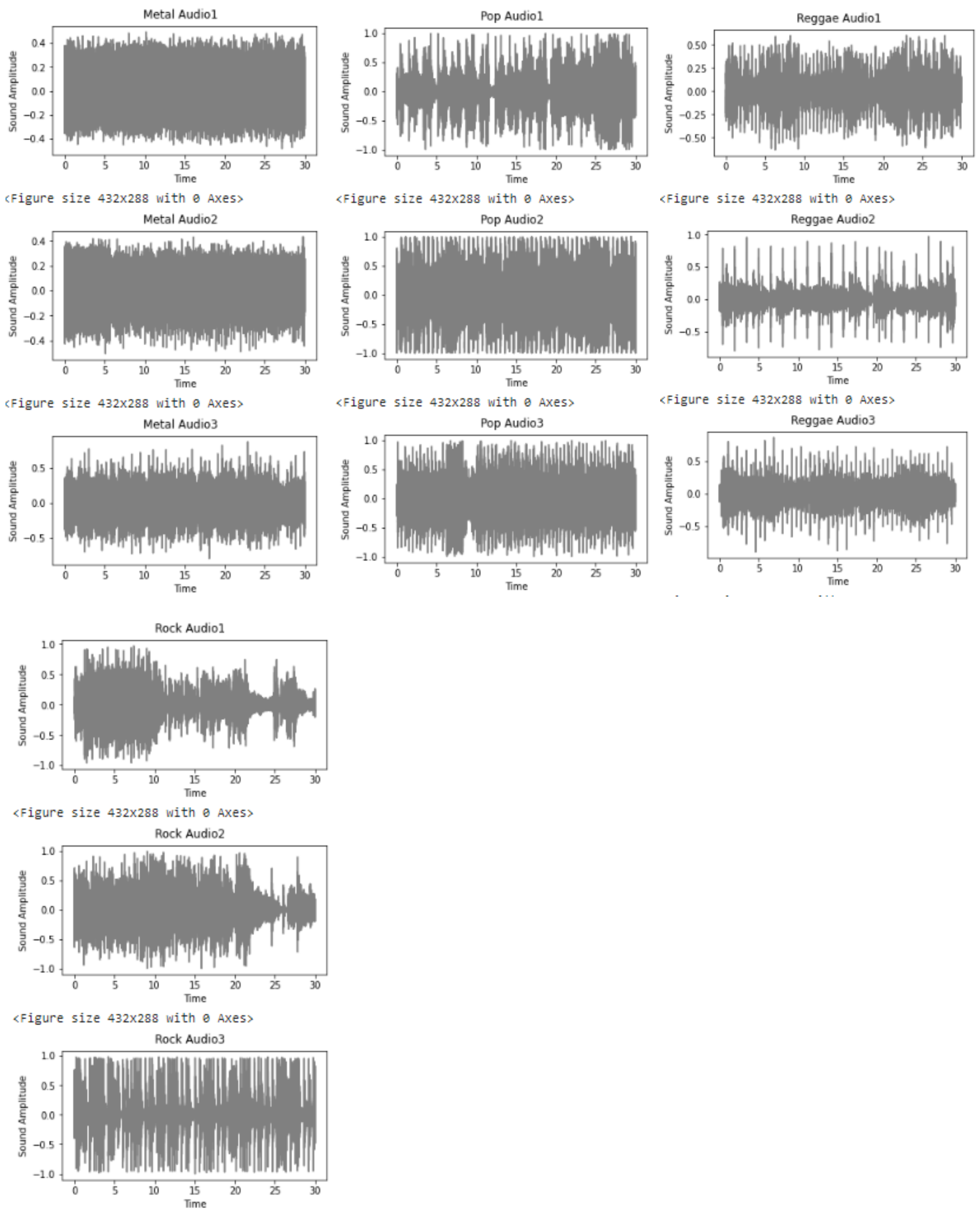

Jazz Audio1
Sound Amplitude
Time

```
<Figure size 432x288 with 0 Axes>
```


Jazz Audio2
Sound Amplitude
Time

```
<Figure size 432x288 with 0 Axes>
```


Jazz Audio3
Sound Amplitude
Time



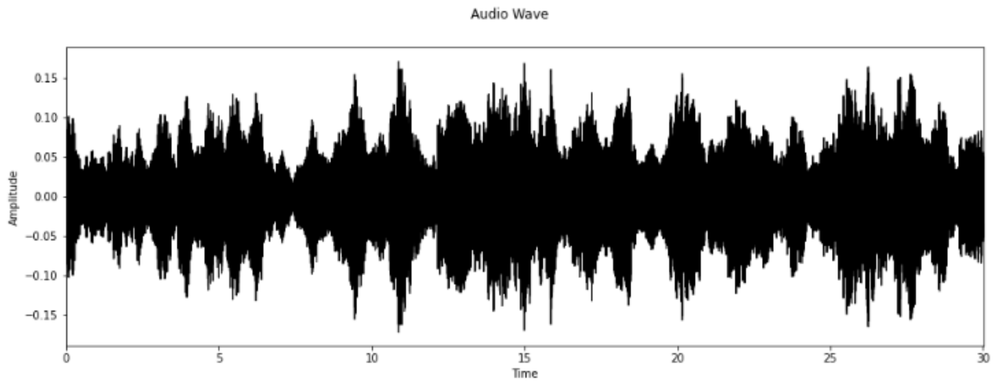
```
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
```

Fig(2): Audio-Waves of genres

1. Audio Wave

   Sample rate defined here is 22050 as the use of librosa is attempted to load the Audio data.
   Signal= Sample rate * Time duration = 22050 * 30
   The plot represents the Amplitude values for each of the signal values. The x axis represents time and Y axis represents Amplitude values.

In the Fig(3) the waveform tends to show a little variation in the amplitude throughout the 30 second time duration.

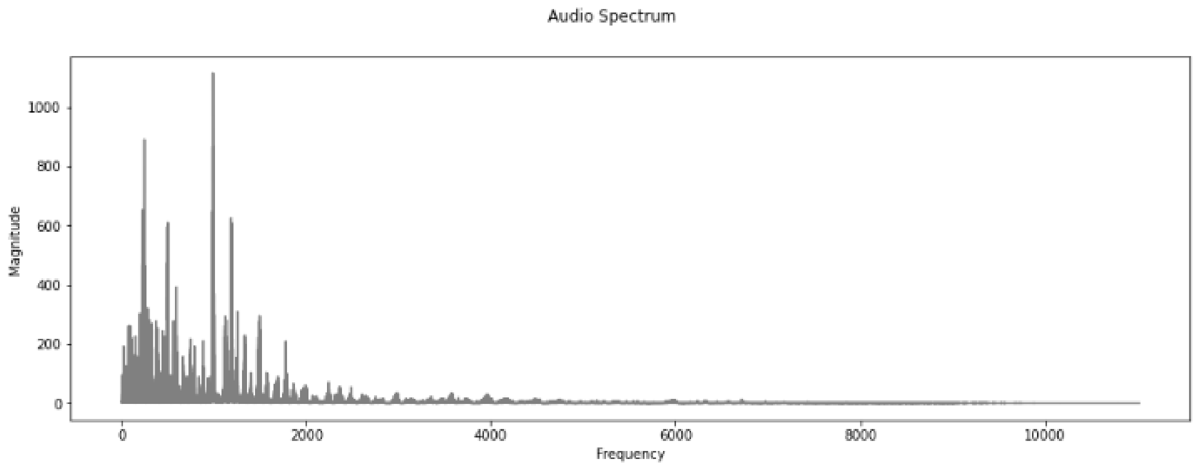


```
<Figure size 432x288 with 0 Axes>
```

Fig(3): Audio wave

2. Audio FFT Spectrum

In order to convert the waveform from the time domain to the frequency domain we need perform the Fast Fourier Transformation (FFT). Use of Numpy is made in order to perform FFT. A one dimensional array is created which has as many values as the total number of samples we have in the waveform. At each of these values we have a complex value out of which we get the magnitude of that value. To get the magnitude values we extract the absolute values of FFT, in simpler words we perform absolute value on the complex values and that's how we obtain the magnitude values.
These magnitudes indicate the contribution of each frequency bin to the overall sound. Here we map the magnitude values with the frequency bins. The use of linspace is made in order to evenly space frequency values in a interval. The magnitude and frequency arrays together tells how much each frequency is contributing to the overall sound. As shown in the plot the x axis represents frequency and while y axis represents magnitude values. In the following Audio spectrum plot, it is observed that most of the energy is concentrated in the lower frequencies and higher the frequencies lower is the energy.



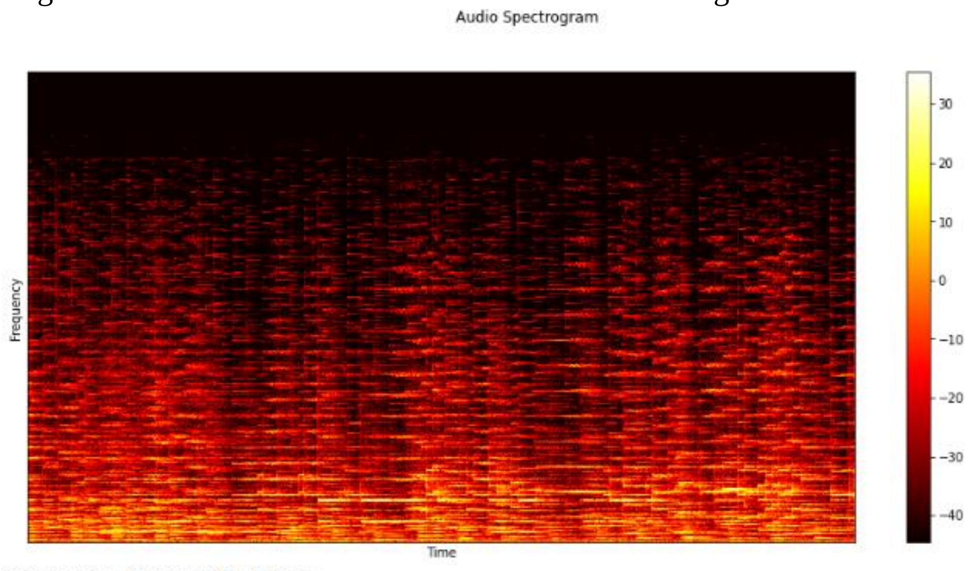
```
<Figure size 432x288 with 0 Axes>
```

Fig(4): Audio Spectrum

3. Audio STFT Spectrogram

The power spectrum is the static snapshot of the whole audio sound. We need to understand how the frequencies are contributing to the overall sound throughout the time domain. To understand this we need STFT (Short Term Fourier Transformation). Which would give the Audio Spectrogram. Spectrogram gives the information about the amplitude as a function of frequency and time. Few values required to create the spectrogram are number of samples per FFT (n_fft) and hop length. n_fft is basically a window which we consider when we are performing a single Fast Fourier Transformation. Hop length is the amount we are shifting each fourier transformation to the right because when performing STFT we slide an interval and at each interval we calculate FFT. And Hop length defines the amount of shifting towards the right. Because we are trying to create a spectrogram, we extract the absolute values of the STFT we obtained. Doing so we extract the magnitude from those complex numbers. Under the librosa we have a function specshow() which enables us to visualize spectrogram like data. The plot is a kind of heat map. In the plot shown below x axis represents Time while the y axis represents Frequency. And the amplitude is expressed through colour bar. The colour represents how the amplitude varies throughout the spectrogram.

Most of the frequencies have very low amplitude and so they contribute very little to the overall sound. While at the lower frequencies some energy bursts can be observed. But in order to have more clear vision through the visualization we calculate the logarithmic values of the spectrogram. To do so we make use of function amplitude_to_db function from librosa. This function takes the values from amplitude values from the spectrogram and converts it into the decibels. And now if compared, the plot 2 gives the better visual representation of the spectrogram. In the plot, the values represented by the colour black represents very very quite sounds in like -40 db while those in the light yellow colour represents the higher intensity frequency values in like 30 db. It is also observed that most of the energy is concentrated in the lower frequencies. The spectrogram remains quite stable throughout time similar to the original sound wave of the audio as seen in the above figure.

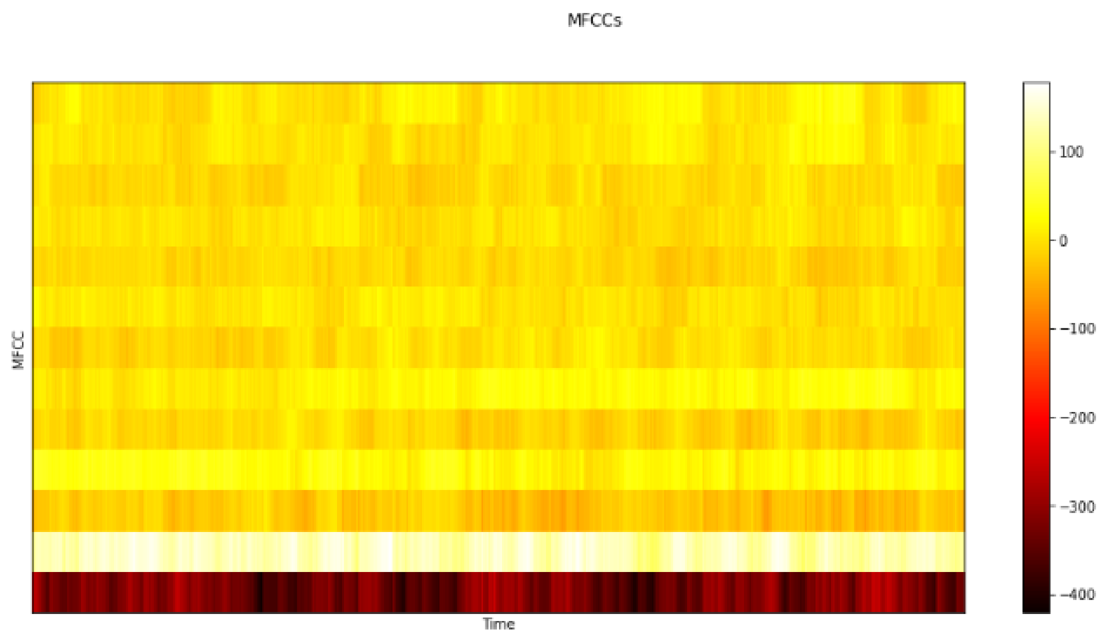


```
<Figure size 432x288 with 0 Axes>
```

Fig(5): Audio Spectrogram

4. MFCCs

In order to extract the MFCCs, the library librosa is used. Librosa.feature.mfcc method is called and certain parameters like the original signal, number of samples per FFT, Hop length, Number of MFCCs are passed to the method (Viswanathan, 2016). The number of MFCCs is the number of the coefficients we want to extract. In my study I extract 13 MFCC features which is most commonly used number of MFCCs for analyzing musical sounds. These MFCC features are used to train the neural networks and classify the audio files as per genres. In the given plot x axis represents Time while y axis represents the MFCCs. Each interval observed on the y axis is a coefficient, it is a MFCC feature. The plot showcases the MFCCs over time. And the MFCC plot seems to appear quite stable.



```
<Figure size 432x288 with 0 Axes>
```

Fig(6): MFCC’s

Methodology

Both Machine Learning and Deep Learning Techniques have been applied to study and compare the performances with respect to the Audio dataset.

1. Machine Learning Technique

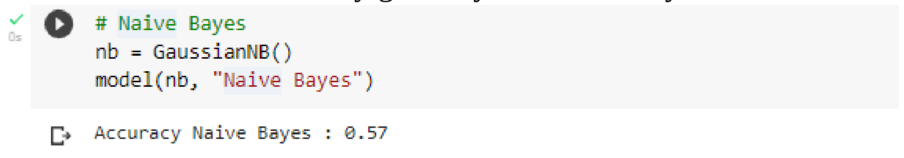
Data Pre-processing

For the machine learning techniques, CSV audio data is used for the classification of the audio files as per genre. The CSV file consists of the various features of each 30-second audio file. All the values in the CSV file are in the float while the length is in integer and the filename and the label column are of datatype object. The file consists of total of 60 columns and 1000 entries. The length of each file is 661794. The file consists of the information about STFT, Spectral centroid, zero-crossing rate, harmony, tempo and MFCCs.

In order to pass the data to the model feature and target, variables are created. The column filename is dropped out of the data. All the variables except the label column are selected as the feature variables. And the label column is selected as the target variable for the model. To prevent the overfitting of the model, the feature variables are normalized using the MinMaxScaler.

The data split for training and testing is 70-30%. 70% of data is used for training and 30% of data for testing.

1. Naïve Bayes
   It uses the Bayes Theorem. It performs the classification on the basis of the probabilities calculated. Based on the given features, it calculates the probability for each target variable. And oy selects the class or the target variable with the largest probability predicted. Hence it is also termed as predictive modelling. To calculate the Gaussian probability density the Gaussian function is used and the probability of the given attribute value is calculated.
   Bayes' Theorem:
   P(A|B)=(P(B|A) * P(A)) / P(B)
   The Classification Accuracy given by the Naive Bayes model is 57%.



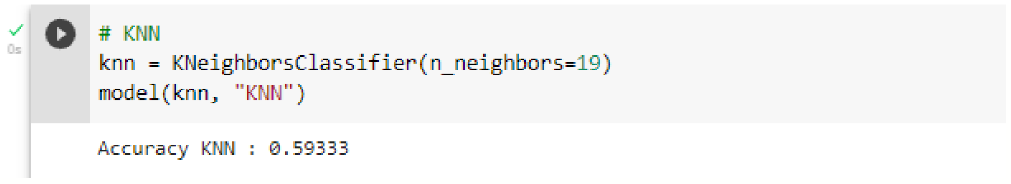
```
# Naive Bayes
nb = GaussianNB()
model(nb, "Naive Bayes")

Accuracy Naive Bayes : 0.57
```

Fig(7): Naïve Bayes

2. KNN

   K-Nearest Neighbours is a supervised learning classification algorithm. The number of maximum votes decides the class into which the data is to be classified. The value of k is decided by the trial and error method. The algorithm classifies the data by computing the distances between the data and all of the reference data points. the k closest data record in the reference data is found by calculating the minimum distance between the data points. There are various distance calculating metrics like Euclidean

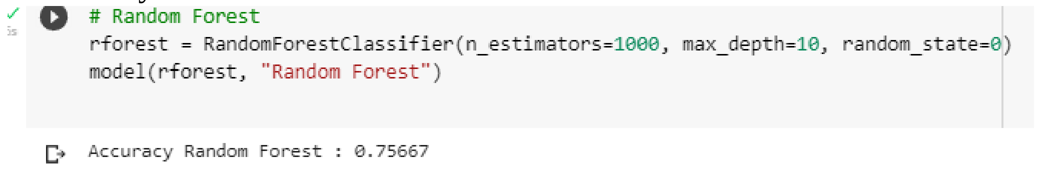
Distance, Manhattan Distance, Hamming distance, Minkowski distance, etc. By default, the `KNeighborsClassifier()` uses the Minkowski distance metric. Whichever is the majority class In the group of k data records is the predicted class of the algorithm. The number of k defined in this model is 19 which is selected on the basis of the trial and error method.



```
# KNN
knn = KNeighborsClassifier(n_neighbors=19)
model(knn, "KNN")

Accuracy KNN : 0.59333
```

Fig(8): KNN

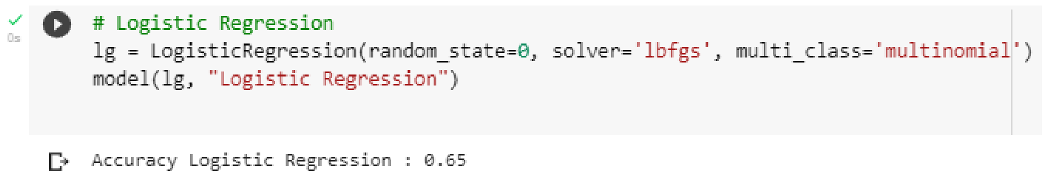
3. Random Forest
   It is the most popular classification algorithm used in the ML field. Random forest algorithm avoids overfitting of the model because it uses multiple trees to train. If the dataset is large then the Random Forest model generates accurate predictions. Also the algorithm is capable of dealing with the missing data values. The algorithm generates multiple Decision Trees during the training phase and as per the majority decision by the trees the final class is selected.



```
# Random Forest
rforest = RandomForestClassifier(n_estimators=1000, max_depth=10, random_state=0)
model(rforest, "Random Forest")

Accuracy Random Forest : 0.75667
```

Fig(9): Random Forest

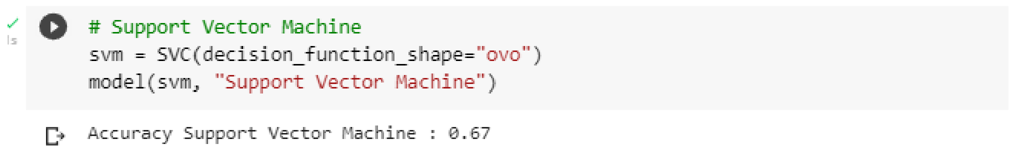
4. Multinomial Logistic Regression
   Logistic Regression is another classification algorithm available which calculates the probability to classify the data. The response variable for this algorithm is always categorical. The algorithm calculates the probabilities for the categorical dependent variables and the class is determined for classification. To use the Logistic regression for multi-classification problems, it needs to define “multinomial” in the function call.



```
# Logistic Regression
lg = LogisticRegression(random_state=0, solver='lbfgs', multi_class='multinomial')
model(lg, "Logistic Regression")

Accuracy Logistic Regression : 0.65
```

Fig(10): Logistic Regression

5. SVM
   Support Vector Machine is another supervised learning algorithm used for classification. SVM performs classification by drawing hyperplanes in such a way that similar kinds of data points are on the same side of the hyperplane. SVM finds the hyperplanes which more accurately classify similar data in such a way that the distance between two kinds of classes is maximum. This maximum distance is termed as margin.



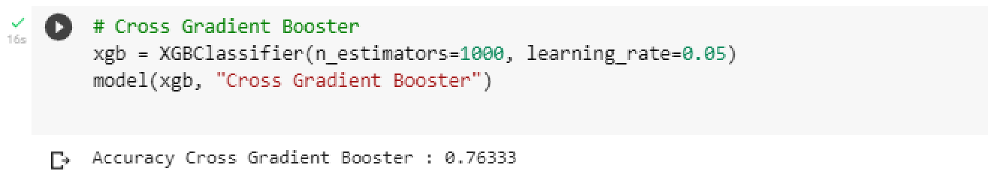
```
# Support Vector Machine
svm = SVC(decision_function_shape="ovo")
model(svm, "Support Vector Machine")
```

```
Accuracy Support Vector Machine : 0.67
```

Fig(11):SVM

6. Cross Gradient Booster

   eXtreme Gradient Boosting is based on gradient boosted decision trees algorithm. XGBClassifier applies a better regularization technique to reduce overfitting. Even with the small dataset, it tends to perform better. It is an ensemble learning algorithm. In the model two hyper tuning parameters are passed. 1000 n_estimators are defined and the learning rate is defined to be 0.05 for the gradient descents.



```
# Cross Gradient Booster
xgb = XGBClassifier(n_estimators=1000, learning_rate=0.05)
model(xgb, "Cross Gradient Booster")
```

```
Accuracy Cross Gradient Booster : 0.76333
```

Fig(12): Gradient Booster

2. Neural Networks

Data Pre-processing

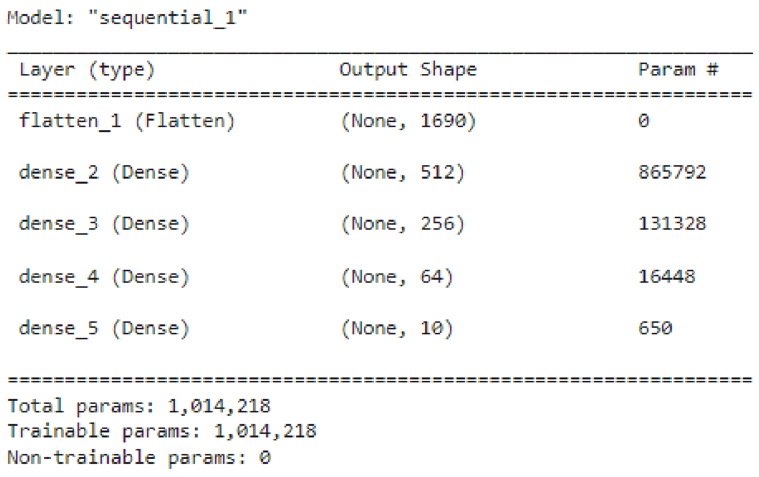
In order to pass the data to the model, it is essential to pre-process the data. By performing the pre-processing of the data, the inputs are extracted. MFCCs are extracted from the music dataset and stored in a JSON file which is used to train the neural networks. To convert and extract the data as per need a user-defined function is generated. A bunch of arguments are passed to this function like the dataset path, and JSON path where the JSON file created will be stored which will be consisting of MFCCs and genre labels. Apart from these two file locations some other arguments like the number of MFCCs, number of FFT, number of segments and hop length values are passed to the function.

The function creates three dictionaries to store the data. The three dictionaries created are mappings, mfcc and labels.Mappings is a list which provides a way to map the different genre labels onto numbers. Mfcc stores mfcc vectors for each segment which is used as training inputs for the network and the labels are the outputs or the targets under among which the audio file is classified.

To loop through all the genres a method os.walk() is used which generates a list. This method returns the list of defined path, directory and file names. Initiating a for loop through os.walk method allows recursive travel through all the data folders from the dataset. To keep the count of the number of iterations a method “enumerate” is used. Also, the semantic labels like classical, blues, rock and others are extracted from the data folder using dirpath.split method. The values extracted are stored into the mappings list. To load and extract the MFCCs, the audio data is loaded using librosa.load method. The MFCC features are extracted at the level of the segments. Hence the sound signal is divided into segments and saved into the mfcc list. For each iteration the loop travels into different genre of the dataset and that genre is stored into the labels list. The dictionary named data consists of these three lists, mappings, mfcc and labels. And this data is stored into the JSON file which is used into the neural network models.

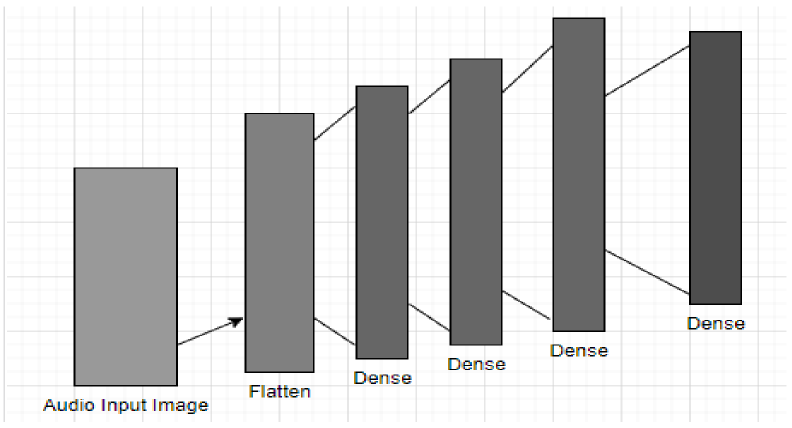
1. ANN

The ANN model consists of a total of five layers. One flattening layer and four fully connected layers. For the layers 2,3 and 4 activation function, ReLU has been applied and as the problem is a classification problem, the last layer uses the softmax activation function. The optimizer used is adam from Keras and the loss function used for the model is "sparse_categorical_crossentropy". Sparse categorical crossentropy produces a category index of the most likely matching category.(***** write about loss function used). The following ANN has been trained with 32 batch size and with 50 epochs.

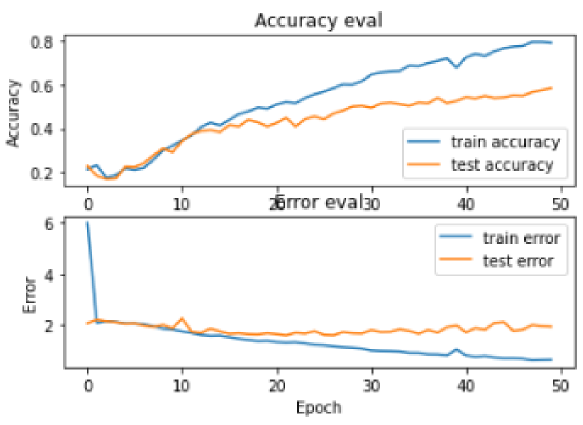


```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_1 (Flatten)         (None, 1690)              0

 dense_2 (Dense)             (None, 512)               865792

 dense_3 (Dense)             (None, 256)               131328

 dense_4 (Dense)             (None, 64)                16448

 dense_5 (Dense)             (None, 10)                650

=================================================================
Total params: 1,014,218
Trainable params: 1,014,218
Non-trainable params: 0
```

Fig(13):ANN model



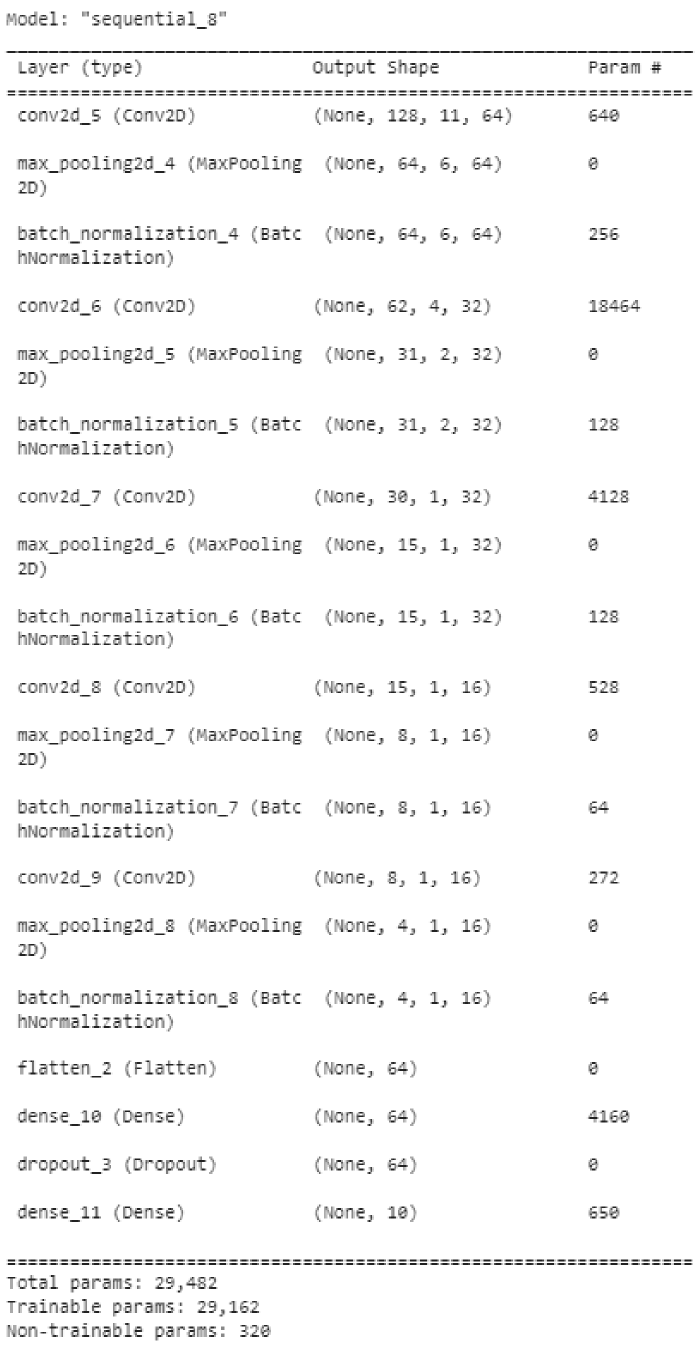
Fig(14):ANN model structure



Fig(15): ANN accuracy

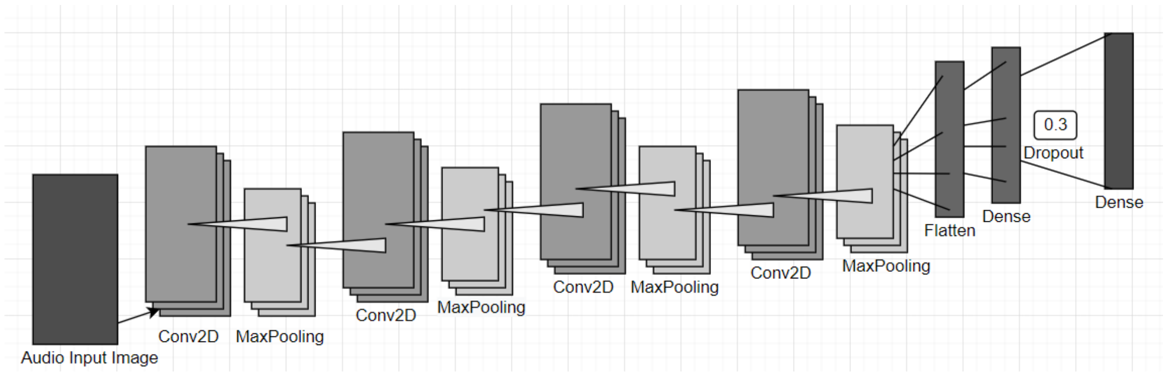
2. CNN

The CNN model consists of four convolutional layers, four max_pooling layers, four batch_normalization layers, one flattening layer, a dropout layer and two fully connected layers. The optimizer used is adam and the loss function used is sparse_categorical_crossentropy.

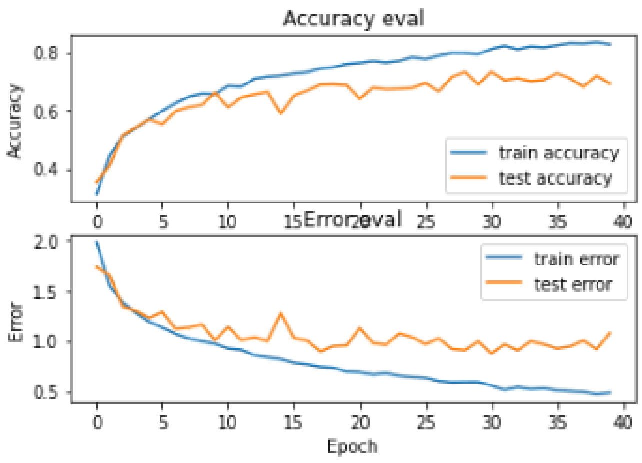


```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_5 (Conv2D)           (None, 128, 11, 64)       640

 max_pooling2d_4 (MaxPooling  (None, 64, 6, 64)        0
 2D)

 batch_normalization_4 (Batc  (None, 64, 6, 64)        256
 hNormalization)

 conv2d_6 (Conv2D)           (None, 62, 4, 32)         18464

 max_pooling2d_5 (MaxPooling  (None, 31, 2, 32)        0
 2D)

 batch_normalization_5 (Batc  (None, 31, 2, 32)        128
 hNormalization)

 conv2d_7 (Conv2D)           (None, 30, 1, 32)         4128

 max_pooling2d_6 (MaxPooling  (None, 15, 1, 32)        0
 2D)

 batch_normalization_6 (Batc  (None, 15, 1, 32)        128
 hNormalization)

 conv2d_8 (Conv2D)           (None, 15, 1, 16)         528

 max_pooling2d_7 (MaxPooling  (None, 8, 1, 16)         0
 2D)

 batch_normalization_7 (Batc  (None, 8, 1, 16)         64
 hNormalization)

 conv2d_9 (Conv2D)           (None, 8, 1, 16)          272

 max_pooling2d_8 (MaxPooling  (None, 4, 1, 16)         0
 2D)

 batch_normalization_8 (Batc  (None, 4, 1, 16)         64
 hNormalization)

 flatten_2 (Flatten)         (None, 64)                0

 dense_10 (Dense)            (None, 64)                4160

 dropout_3 (Dropout)         (None, 64)                0

 dense_11 (Dense)            (None, 10)                650

=================================================================
Total params: 29,482
Trainable params: 29,162
Non-trainable params: 320
```

Fig(16):CNN model



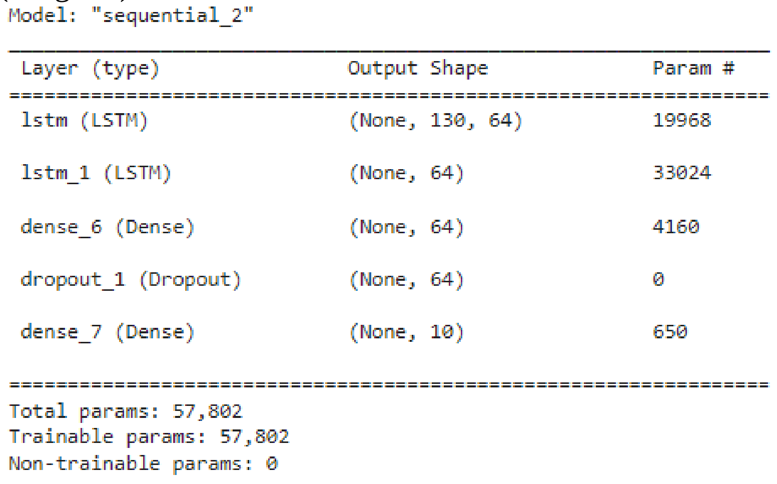
Fig(17): CNN model accuracy



Fig(18): CNN accuracy

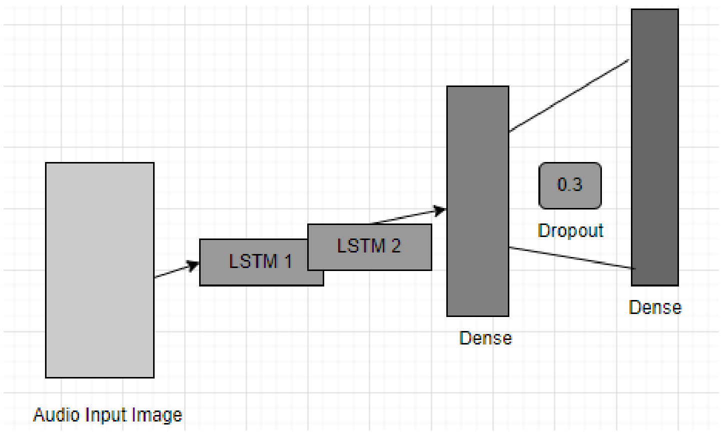
3. RNN

The RNN model is applied along with the LSTM layers. The model consists of two LSTM layers, a dropout layer and two fully connected layers. The activation function used on the dense layer is ReLU and last layer uses the softmax activation function. The optimizer used is adam and loss function used is sparse_categorical_crossentropy.
(Diagram)

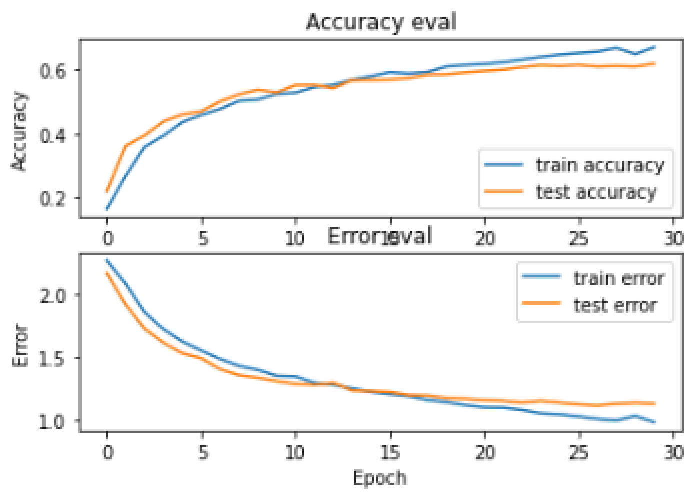


```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 130, 64)           19968

 lstm_1 (LSTM)               (None, 64)                33024

 dense_6 (Dense)             (None, 64)                4160

 dropout_1 (Dropout)         (None, 64)                0

 dense_7 (Dense)             (None, 10)                650

=================================================================
Total params: 57,802
Trainable params: 57,802
Non-trainable params: 0
```

Fig(19): RNN model



Fig(20): RNN model structure



Fig(21): RNN accuracy

Results

The music audio CSV dataset has been successfully applied to six Machine Learning algorithms and the sound dataset of the Music audio has been pre-processed and applied to train and test on the Neural Networks. While working on the numerical data of the Music audio the best performance to classify the files was given by Gradient boost classifier. When working with the sound data it is found that CNN does a better classification than ANN and RNN.

Performance measure of ML Algorithms

ML model	Accuracy
Naïve Bayes	57.0%
KNN	59.3%
Random Forest	75.6%
Logistic Regression	65.0%
SVM	67.0%
Cross Gradient Booster	76.3%

Table(1): ML model performances

Performance measure of Neural Networks

Model	Train loss	Test loss	Train Accuracy	Test Accuracy
ANN	74.8%	1.90	73.7%	58.2%
RNN	94.4%	1.12	69.0%	60.4%
CNN	32.2%	0.74	88.4%	76.7%

Table(2): DL model performances

Discussion

Among all the ML algorithms applied to the CSV audio dataset, the Random forest and gradient boost classifier had given better accuracy in the comparison. Both of these ML algorithms use decision trees. These ensemble ML algorithms avoid overfitting of the data. Gradient boost classifier is an advanced version of Random forest. Hence Gradient boost classifier performs better than the Random Forest algorithm. Gradient boost classifier sequentially reduces error from the data and the weighted average of various trees prediction gives the final class prediction. In gradient boosting all the trees work on all the data points. Random forest works parallelly on the samples while Gradient boosting works sequentially on data and next trees focus on residual errors and do prediction of the class. But both these are only capable to work on tabular data of audio. In order to work with original audio data more advanced version of machine learning is required. This is where deep learning comes into the picture. Deep learning is capable of working with images, texts, sounds or videos.

When dealing with the neural networks for audio data classification, one needs to understand the features of the audio data to study their patterns (Rege & Sindal, 2021). The spectrogram is the visual representation of all the frequencies over time. It was found that the CNN model can outperform RNN and ANN models when working with image datasets. CNN is specially designed to work upon image datasets. The inputs passed to the neural network are also a visual representation of audio data. In order to classify the audio data, the audio data was converted into spectrograms which are two-dimensional visual representations of frequency varying over time. CNN can detect key features from an image to distinguish it from other images. ANN works well over text or tabular data while RNN works well for sequential data. CNN's can discriminate spectro-temporal patterns. They are able to capture patterns across time and frequency domains for given input spectrogram. CNNs have an edge over traditional mell frequency substrate coefficients MFCCs because these Convolutional neural networks are still able to make distinctions even when sound is masked in time and frequency. In order to get more accurate results from the model, more data is required on which the model can be trained. Hence data augmentation plays a major role to improve the model's accuracy.

Conclusion

The traditional ML algorithms can work fine if tabular audio data needs to be classified like a CSV file. But in real life, it would be a hassle to create and update tabular data for every new audio. Every day nearly about 60 thousand new tracks are added to Spotify and to classify the tracks as per their genres using ML algorithms would require time and effort. Audio files have a complex structure. Before classifying the audio data, one will have to extract the audio features and store them into a tabular structured file and then pass it to the model. While using deep learning techniques one can directly extract audio features from an audio file into a 2D visual representation and perform classification. To achieve more accurate results, the model would require a large amount of training data.

References

-Journal Article

1. Bhatia, J. K., Singh, R. D., & Kumar, S. (2021). Music Genre Classification. *2021 5th International Conference on Information Systems and Computer Networks, ISCON 2021*, *November*. https://doi.org/10.1109/ISCON52037.2021.9702303

2. Ceylan, H. C., Hardalaç, N., Kara, A. C., & Hardalaç, F. (2021). Automatic Music Genre Classification and Its Relation with Music Education. *World Journal of Education*, *11*(2), 36. https://doi.org/10.5430/wje.v11n2p36

3. Li, H., Xue, S., & Zhang, J. (2018). *Combining CNN and Classical Algorithms for Music Genre Classification*. 1–6. http://cs229.stanford.edu/proj2018/report/19.pdf

4. Rege, A., & Sindal, R. (2021). Audio classification for music information retrieval of Hindustani vocal music. *Indonesian Journal of Electrical Engineering and Computer Science*, *24*(3), 1481–1490. https://doi.org/10.11591/ijeecs.v24.i3.pp1481-1490

5. Tzanetakis, G., & Cook, P. (2002). Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, *10*(5), 293–302. https://doi.org/10.1109/TSA.2002.800560

6. Viswanathan, A. P. (2016). Music Genre Classification. *International Journal Of Engineering And Computer Science*. https://doi.org/10.18535/ijecs/v4i10.38

--Online document

https://www.grandviewresearch.com/industry-analysis/music-streaming-market

Annexure

Appendix