Music Genre Classification using Spectral Analysis and Random Forest Classifier

Iman Iraei (Student ID: 40204784) Mostafa Abolfazli (Student ID: 2141504)

Fatemeh Akbari (Student ID: 40213781) Hassan Mokari (Student ID: 2199093)

https://github.com/iman10115/Machine-Learning-Project-Fall-2022

***Abstract*—Musical genres are categorical labels created by humans to characterize pieces of music. A musical genre is characterized by the common characteristics shared by its members. These characteristics typically are related to the instrumentation, rhythmic structure, and harmonic content of the music. Therefore, in this project, we will implement our method using multi-class SVM to classify miscellaneous musical genres such as Blues, Classical, Country, Disco, Hip-hop, Jazz, Metal, Pop, Reggae, and Rock, we rely purely on Mel Frequency Cepstral Coefficients (MFCC) to characterize the dataset. Once the features are extracted, different standard machine learning techniques which are independent of the specific application area will be used as classifiers. Hence, the main objective of this project is to make better musical genres prediction as well as comparing predictive performance of different machine learning classifiers including Support Vector Machine(SVM), Random Forest, k-nearest Neighbor (KNN) and Gradient Boosting to investigate the results. At the end, regarding to measure the success of our study, we will make genre confusion matrix and also use an accuracy criterion in order to evaluate the results for music genre classification.**

***Index Terms*—Music Genre, spectral analysis, multiclass classification, non-stationary signal, support vector machine**

I. Introduction

Music genre classification forms a basic step for building a strong recommendation system. The idea behind this project is to see how to handle sound files in python, compute sound and audio features from them, run Machine Learning Algorithms on them, and see the results. It aims to predict the genre using an audio signal as its input.

The objective of automating the music classification is to make the selection of songs quick and less cumbersome. If one has to manually classify the songs or music, one has to listen to a whole lot of songs and then select the genre. This is not only time-consuming but also difficult.

The basis of any type of automatic audio analysis system is the extraction of feature vectors. A large number of different feature sets, mainly originating from the area of speech recognition, have been proposed to represent audio signals. Typically, they are based on some form of time-frequency representation. Due to the fact that audio signals are Non-stationary signals, they can be modeled as the product of a narrow bandwidth low-pass process modulating a higher bandwidth carrier. [1] In order to achieve the best performance of the classifiers, first we need to select the best features as the inputs of the classification algorithms. When working with audio signals, MFCCs are certainly the most important features due to the non-stationary attributes of an audio signal. These coefficients are perceptually motivated features based on the Short-Time Fourier Transform (STFT). After taking the log-amplitude of the magnitude spectrum of a signal, the FFT bins are grouped and smoothed according to the perceptually motivated Mel-frequency scaling. [2] Finally, a discrete cosine transform is performed in order to decorrelate the resulting feature vectors.

An audio signal is constantly changing, so to simplify things we assume that on short time scales the audio signal doesn't change much (when we say it doesn't change, we mean statistically stationary, obviously the samples are constantly changing on even short time scales). This is why we frame the signal into 20-40ms frames. If the frame is much shorter, we don't have enough samples to get a reliable spectral estimate, if it is longer the signal changes too much throughout the frame. The next step is to calculate the power spectrum of each frame. For this reason, we take clumps of periodogram bins and sum them up to get an idea of how much energy exists in various frequency regions. This is performed by our Mel filterbank. The first filter is very narrow and gives an indication of how much energy exists near 0 Hertz. As the frequencies get higher our filters get wider as we become less concerned about variations. The final step is to compute the DCT of the log filterbank energies. There are 2 main reasons this is performed. Because the filterbanks are all overlapping, the filterbank energies are quite correlated with each other. The DCT decorrelates the energies which means diagonal covariance matrices can be used to model the features multi-class SVM.

II. Related Work

Some related works are reviewed in this section. In recent years, there are several series of studies related to music genre classification using machine learning algorithms.
In the research conducted by Pelchat and Craig M. [3] The GTZAN dataset's songs were categorised into seven genres. The stereo channels were then combined into one mono channel, and the music data was converted into a spectrogram using the SoX (Sound eXchange) command-line music application utility, which was then sliced into 128x128 pixel images, and the labelled spectrogram was used as inputs to the dataset, which was split into 70% training data, 20% validation data, and 10% test data. All of the weights were now initialised using the Xavier initialization method. The first four layers are convolutional layers with a kernel size of 2x2 and a stride of two, followed by a max pooling layer. Following the first four layers is a fully connected layer in which each output of the previous layer is fed into each input of the fully connected layer. This yields a vector of 1024 numbers. After that, a SoftMax layer is applied to generate seven outputs, one for each genre. The CNN implementation appeared to be overfit, as the accuracy for the training data was 97% versus 47% for the test data.
Another interesting series of studies by K. Meenakshi [4] were used to train the system by categorising the music database into different genres. After that, each song must go through a pre-processing stage. Feature Vector Extraction is performed in Python using the librosa package, also known as MFCC. The Mel Scale Filtering is then performed to obtain the Mel Frequency Spectrum by [4]. They obtained two types of feature vectors: Mel Spectrum with 128 coefficients and MFCC coefficients. ConvNet architectures are built using three types of layers: Convolutional Layer, Pooling Layer, and Fully Connected Layer. The database thus obtained is the MFCC, with a genre array size of 10 arrays. The input consists of 1000 songs with ten labels. The Anaconda Python package was used for the evaluation. The learning accuracy of the Mel Spec feature vector and the MFCC feature vector was found to be 76% and 47%, respectively.
Research conducted by Nirmal M R [5]. They did use a spectrogram, which is a visual representation of a signal's frequency spectrum as it varies over time. A spectrogram is a signal's Short Time Fourier Transform (STFT). Spectrograms are graphs that represent data in both the time and frequency domains. Two models are used to implement the convolutional neural network, and their results are User-defined sequential CNN model and Pre-trained ConvNet (MobileNet). The classification accuracy of the user-defined CNN model and MobileNet was 40% and 67%, respectively. Python 3 was used in the LINUX operating system. The deep learning model used in [5] was built-in Python Keras framework.

The methods used in [6] the stages of the proposed method were used in their research. Data collection and selection, pre-processing, feature extraction and selection, classification, evaluation, and measurement are the following methodologies used in [6]. This research makes use of the Spotify music dataset, which contains 228,159 songs across 26 genres and 18 features. To process data structures and perform data analysis, the Python programming language is used, along with the Python Data Analysis Library (PANDAS). Later on, this genre feature will be used to classify the target. This is done especially for the learning process that uses the SVM classifier. Using hyperparameter, the SVM-RBF classification was carried out by searching the grid for the best results. Kfold cross-validation with free random conditions and a comparison of 80% of training data and 20% of test data. The accuracy for the SVM, KNN, and NB was 80%, 77.18%, and 76.08%, respectively.

With the recent success of deep neural networks, a number of studies apply these techniques to speech and other forms of audio data [7] [6]. Representing audio in the time domain for input to neural networks is not very straight-forward because of the high sampling rate of audio signals. However, it has been addressed in Van Den Oord et al. [6] for audio generation tasks. A common alternative representation is the spectrogram of a signal which captures both time and frequency information. Spectrograms can be considered as images and used to train convolutional neural networks (CNNs). [8] A CNN was developed to predict the music genre using the raw MFCC matrix as input in Li et al. [9]. In Lidy and Schindler [10], a constant Q-transform (CQT) spectrogram was provided as input to the CNN to achieve the same task.

III. PROPOSED METHODOLOGY

A. Feature Extraction

The librosa module in Python is used for Feature Vector Extraction. Each audio file is extracted and the vector feature is calculated. By recording approximately, the type of log energy spectrum on the Mel-frequency scale, MFCCs incorporate timbral features of the music signal.

- **Pre-emphasis:** Pre-emphasis refers to filtering that emphasizes the higher frequencies. For voiced sounds, the glottal source has an approximately 12 dB/octave slope [11]. However, when the acoustic energy radiates from the lips, this causes a roughly +6 dB/octave boost to the spectrum.Therefore, pre-emphasis removes some of the glottal effects from the vocal tract parameters. The most commonly used pre-emphasis filter is given by the following transfer function:

$$H(z) = 1 - bz^{-1} \tag{1}$$

  where the value of b controls the slope of the filter and is usually between 0.4 and 1.0 [11].

- **Frame blocking and windowing:** The speech signal is a slowly time-varying or quasi-stationary signal. For stable

acoustic characteristics, speech needs to be examined over a sufficiently short period of time. Short-term spectral measurements are typically carried out over 20ms windows, and advanced every 10ms [12] [13]. The purpose of the overlapping analysis is that each speech sound of the input sequence would be approximately centered at some frame. On each frame, a window is applied to taper the signal towards the frame boundaries. Generally, Hanning or Hamming windows are used [11]. This is done to enhance the harmonics, smooth the edges, and to reduce the edge effect while taking the DFT on the signal.

- **DFT spectrum:** Each windowed frame is converted into magnitude spectrum by applying DFT.

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{\frac{-j2\pi nk}{N}}; \quad 0 \leq k \leq N-1 \qquad (2)$$

where N is the number of points used to compute the DFT.

- **Mel spectrum:** Mel spectrum is computed by passing the Fourier transformed signal through a set of band-pass filters known as Mel-filter bank. AMel is a unit of measure based on the human ears perceived frequency. The Mel scale is approximately a linear frequency spacing below 1 kHz and a logarithmic spacing above 1 kHz [14]. The approximation of Mel from physical frequency can be expressed as

$$f_{Mel} = 2595 \log_{10}(1 + \frac{f}{700}) \qquad (3)$$

where f denotes the physical frequency in Hz, and f_{Mel} denotes the perceived frequency [12].
For MFCC computation, filter banks are generally implemented in frequency domain. The center frequencies of the filters are normally evenly spaced on the frequency axis. However, in order to mimic the human ears perception, the warped axis, according to the nonlinear function given in Eq. (3), is implemented. The most commonly used filter shaper is triangular, and in some cases the Hanning filter can be found [11]. The triangular filter banks with Mel frequency warping is given in Fig. 1.
The Mel spectrum of the magnitude spectrum $X(k)$ is computed by multiplying the magnitude spectrum by each of the of the triangular Mel weighting filters.

$$s(m) = \sum_{k=0}^{N-1} |X(k)|^2 H_m(k); \quad 0 \leq m \leq M-1 \qquad (4)$$

where M is total number of triangular Mel weighting filters [16] [17]. $H_m(k)$ is the weight given to the k^{th} energy spectrum bin contributing to the mth output band and is expressed as:

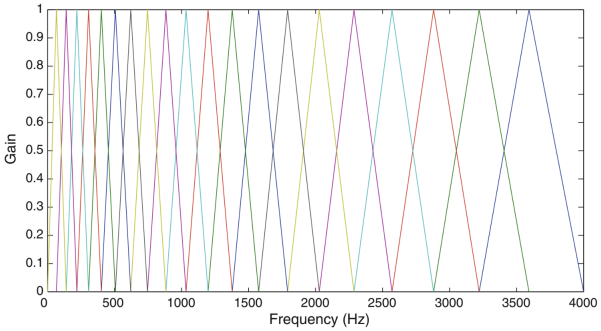

Fig. 1. Mel-filter bank. The image taken from [15]

$$H_m(k) = \begin{cases} 0 & k < f(m-1) \\ \frac{2(k-f(m-1)}{f(m)-f(m-1)} & f(m-1) \leq k \leq f(m) \\ \frac{2(f(m+1)-k}{f(m+1)-f(m)} & f(m) < k \leq f(m+1) \\ 0 & k > f(m+1) \end{cases} \qquad (5)$$

with m ranging from 0 to $M-1$.

- **Discrete cosine transform (DCT):** Since the vocal tract is smooth, the energy levels in adjacent bands tend to be correlated. The DCT is applied to the transformed Mel frequency coefficients produces a set of cepstral coefficients. Prior to computing DCT, the Mel spectrum is usually represented on a log scale. Since most of the signal information is represented by the first few MFCC coefficients, the system can be made robust by extracting only those coefficients ignoring or truncating higher order DCT components [11]. Finally, MFCC is calculated as [11]:

$$s(m) = \sum_{m=0}^{M-1} \log_{10} s(m) \cos(\frac{\pi n(m-0.5)}{M}) \qquad (6)$$

where $n = 0, 1, 2, ..., C-1$ and $c(n)$ are the cepstral coefficients, and C is the number of $MFCCs$. Traditional MFCC systems use only 8–13 cepstral coefficients. The zeroth coefficient is often excluded since it represents the average log-energy of the input signal, which only carries little speaker-specific information.
- **Dynamic MFCC features:** The cepstral coefficients are usually referred to as static features, since they only contain information from a given frame. The extra information about the temporal dynamics of the signal is obtained by computing first and second derivatives of cepstral coefficients [18] [19] [20]. The commonly used definition for computing dynamic parameter is [18]:

$$\Delta c_m(n) = \frac{\sum_{i=-T}^{T} k_i c_m(n+1)}{\sum_{i=-T}^{T} |i|} \qquad (7)$$

where $c_m(n)$ denotes the m^th feature for the n^th time frame, k_i is the i^{th} weight, and T is the number of successive frames used for computation. Generally T is taken as 2. The delta–delta coefficients are computed by taking the first-order derivative of the delta coefficients.

B. Dataset

In this project, we use GTZAN genre collection dataset with different numbers of audio files and classes. It consists of 1000 audio files each having 30 seconds duration. There are 10 classes (10 music genres) each containing 100 audio tracks. Each track is in .wav format. It contains audio files of the following 10 genres: Blues, Classical, Country, Disco, Hiphop, Jazz, Metal, Pop, Reggae and Rock. So, our data is mainly speech signals which are converted to a lot of numerical feature vectors in time domain. When these 1000 speech signals are converted to MFCC features, consequently we have 1000 feature vectors from which 70% are as training samples and the remaining 30% of them are our testing samples. All the computational including training process can be implemented on a laptop.

For normalization of our data, we used MinMaxscaler which transforms features by scaling each feature to a specific range. This estimator scales and translates each feature individually such that it is in the given range on the training set between zero and one. The MinMaxscaler is a type of scaler that scales the minimum and maximum values to be 0 and 1 respectively. Also, we need to split a dataset into train and test sets to evaluate how well our machine learning model performs. The train set is used to fit the model, and the statistics of the train set are known. The second set is called the test data set, this set is solely used for predictions. A commonly used ratio is 80:20, which means 80% of the data is for training and 20% for testing. There does not seem to be clear guidance on what ratio is best or optimal for a given dataset.

For feature extraction, in the first step we need to have good representation of our signal. As we mention before the audio data have a time domain representation of a regular signal. So, if we illustrate it, we have lots of values that fluctuate along with the x-axis. As we can see the main signal is shown in Fig. 2 in time domain that it is known as our raw data.

Spectrograms are calculated from the time domain using

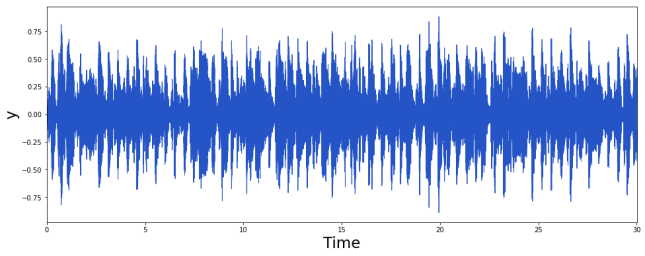

Fig. 2. Sound wave of blue 0 signal in time domain

Fourier Transform. When the audio is sampled (in the time domain), it is segmented into several overlapping windows. Then, the Short-time Fourier Transform (STFT) is used to calculate the frequency spectrum for each window and each window represents a vertical line in the image. These parts are put together side by side. This process is called windowing. The STFT of an audio file is shown in Fig. 3.

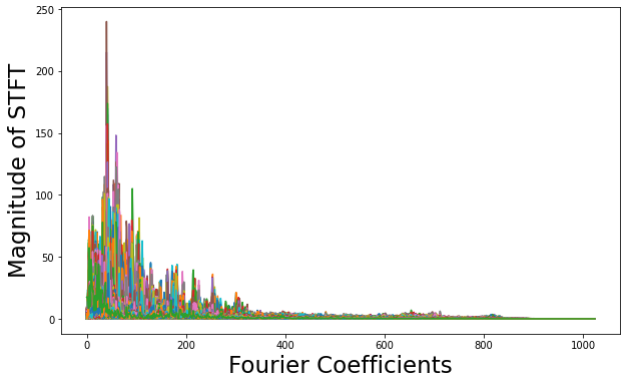

Fig. 3. STFT of each window in frequency domain

The process of creating a spectrogram through a digital process essentially involves the computation of the squared magnitude of the STFT of the signal for a particular window width. The spectrogram of an audio file is shown in Fig. 4, where pink represents high amplitude and blue represents low amplitude.

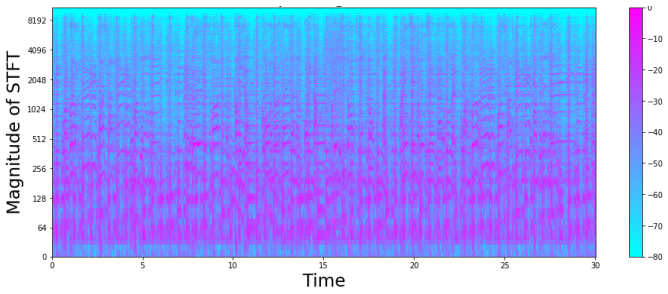

Fig. 4. Spectrum of the signal. The magnitude of a signal versus time representation

In fact, the spectrogram is the visual representation of the strength of a signal over time with different frequencies present at each time step. In the case of audio, it is also known as sonographs or voicegrams. There are three dimensions in a spectrogram, two are frequency (y-axis) and time(x-axis), and the third dimension is amplitude.

In a Mel-Spectrogram, the spectrogram is converted to mel-scale using mel filter-banks. After computing a spectrogram, we map the frequency (y-axis) of the spectrogram to the mel-scale to form a Mel-Spectrogram. The Mel-Spectrogram of an audio file is shown in Fig. 5.

MFCC is Mel Frequency Cepstral Coefficients. They are a small set of features that precisely describe the complete shape of a spectral envelope. The MFCC uses the MEL scale to divide the frequency band into sub-bands. Then, the Cepstral Coefficients are extracted by computing the Discrete

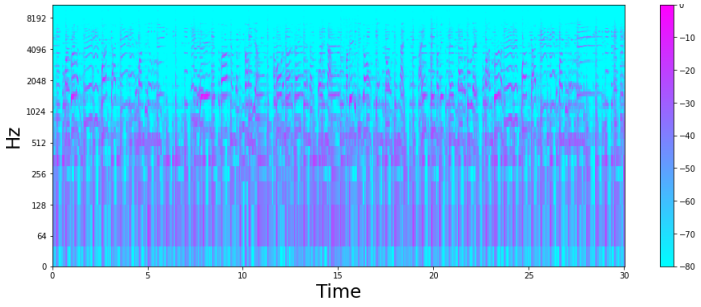

Fig. 5. Mel spectrum of a Blue signal

Cosine Transform (DCT). In a way, MFCCs compress the data of a Mel-Spectrogram. DCTs are popular for easy image compression. The MFCC of an audio file is shown in Fig. 6. A correlation matrix is simply a table which displays the

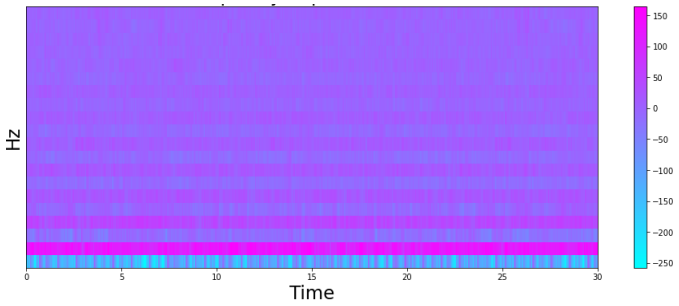

Fig. 6. Mel-Frequency Cepstral Coefficients of Blue signal

correlation coefficients for different variables. The matrix depicts the correlation between all the possible pairs of values in a table. It is a powerful tool to summarize a large dataset and to identify and visualize patterns in the given data. In this project, we consider only mean value of different features for our correlation matrix. The correlation matrix is shown in Fig. 7.

Moreover, we used a Box Plots to show distributions of our numeric data values and compare them between multiple groups. They are built to provide high-level information at a glance, offering general information different classes' symmetry, skew, variance, and outliers. A box plot shows the distribution of quantitative data in a way that facilitates comparisons between variables or across levels of a categorical variable.

C. Machine Learning Models

This section provides a brief overview of the four machine learning classifiers adopted in this study.

- **Random Forest (RF):** Random Forest is an ensemble learner that combines the prediction from a pre-specified number of decision trees. It works on the integration of two main principles: 1) Each decision tree is trained with only a subset of the training samples which is known as bootstrap aggregation (or bagging) 2) Each decision tree

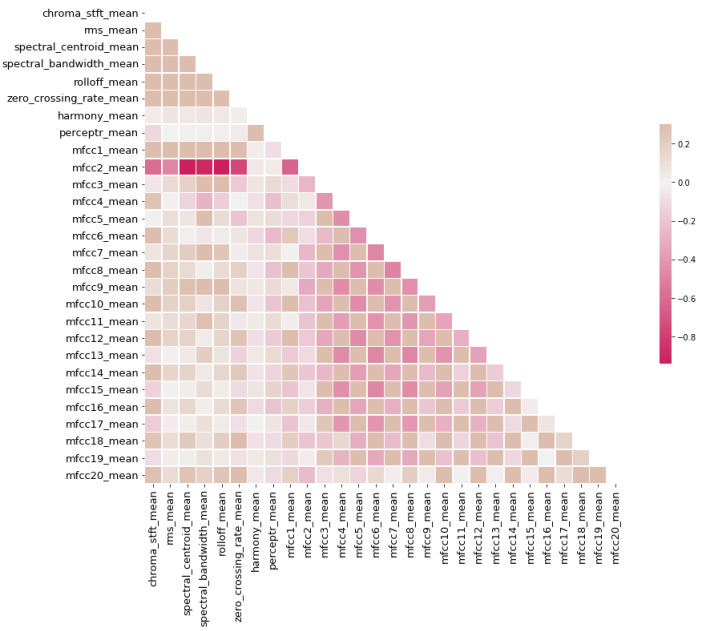

Fig. 7. Correlation Heatmap for the mean variables

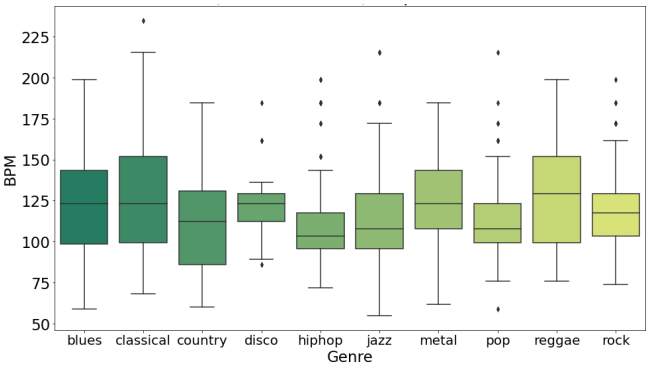

Fig. 8. Beats per minute (BPM) boxplot for each music genre

is required to make its prediction using only a random subset of the features [21]. The final predicted class of the RF is determined based on the majority vote from the individual classifiers.

- **Logistic Regression (LR):** This linear classifier is generally used for binary classification tasks. For this multi-class classification task, the LR is implemented as a one-vs-rest method. That is, 10 separate binary classifiers are trained. During test time, the class with the highest probability from among the 10 classifiers is chosen as the predicted class.

- **Support Vector Machines (SVM):** SVMs transform the original input data into a high dimensional space using a kernel trick [22]. The transformed data can be linearly separated using a hyperplane. The optimal hyperplane maximizes the margin. In this study, a radial

basis function (RBF) kernel is used to train the SVM because such a kernel would be required to address this non-linear problem. Similar to the logistic regression setting discussed above, the SVM is also implemented as a one-vs-rest classification task.

- **K-Nearest Neighbors (KNN):** The k-nearest neighbors (KNN) [23] algorithm is a simple, supervised machine learning algorithm that can be used to solve both classification and regression problems. The k-nearest neighbors algorithm, also known as KNN is a non-parametric classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point.

- **Gradient Boosting:** XGBoost is a scalable ensemble technique based on gradient boosting that has demonstrated to be a reliable and efficient machine learning challenge solver [24]. Boosting algorithms combine weak learners into a strong learner in an iterative way [25]. Gradient boosting is a boosting-like algorithm for regression [26]. Given a training dataset, the goal of gradient boosting is to find an approximation, $\hat{F}(x)$ of the function $F(x)$, which maps instances x to their output values y, by minimizing the expected value of a given loss function, $L(y, F(x))$. Gradient boosting builds an additive approximation of $F(x)$ as a weighted sum of functions.

D. Hyperparameters Optimization

Grid search is a process that searches exhaustively through a manually specified subset of the hyperparameter space of the targeted algorithm. Random search, on the other hand, selects a value for each hyperparameter independently using a probability distribution. So, grid search is a tuning technique that attempts to compute the optimum values of hyperparameters. It is an exhaustive search that is performed on the specific parameter values of a model. Grid search exercise can save us time, effort and resources.
For tunning the hyperparameters, one method is to try out different values and then pick the value that gives the best score. If we had to select the values for two or more parameters, we would evaluate all combinations of the sets of values thus forming a grid of values. Therefore, grid-search is used to find the optimal hyperparameters of a model which results in the most 'accurate' predictions.
Hence, in our project we use grid search for tunning hyperparameters of each classifier.
As for SVM classifier we have 3 hyperparameters that need to find the optimal values for each of them and they are kernel, regularization parameter and Gamma. The main function of the kernel is to take low dimensional input space and transform it into a higher-dimensional space. It is mostly useful in non-linear separation problem. Regularisation or C is the penalty parameter, which represents misclassification or error term. The misclassification or error term tells the SVM optimisation how much error is bearable. This is how you can control the trade-off between decision boundary and misclassification term. Gamma can defines how far influences the calculation of plausible line of separation.
In order to the best performance of Random Forest as our second classifier, there are 6 hyperparameters that need to be tunned and they are bootstrap, max_depth, max_features, min_samples_leaf, min_samples_split and n_estimators, respectively. n_estimators determine the number of decision trees in the forest. max_depth defines the maximum number of branches in each decision tree. Regarding k-Nearset Neighbor classifier we have used n-neighbors and algorithm as three option ball tree, kd tree, and brute.
As for tuning the hyperparameters of xgboost classifier, we use learning rate, max depth, gamma, booster and n_estimator. The learning rate shrinks the contribution of each tree. There is a trade-off between the learning rate and n_estimators. The n_estimators is the number of boosting stages to perform. Gradient boosting is fairly robust to over-fitting so a large number usually results in better performance. Also, the maximum depth parameter limits the number of nodes in the tree. we tune this parameter for the best performance.
K-fold Cross-Validation is when the dataset is split into a K number of folds and is used to evaluate the model's ability when given new data. K refers to the number of groups the data sample is split into. Actually, it is used to estimate the performance of machine learning models when making predictions on data not used during training. This procedure can be used both when optimizing the hyperparameters of a model on a dataset, and when comparing and selecting a model for the dataset. During doing the grid search procedure we use 4-fold cross validation. Nevertheless, the K-Fold class can be used directly in order to split up a dataset prior to modeling such that all models will use the same data splits.

IV. Evaluation and Results

In this project, we will achieve music genre classification using machine learning algorithms through a variety of steps. For model creation, we are planning to use the Scikit-learn and PyTorch library. Using the proposed methodology, we hope to present a comparative study of how the various models perform.

A. Evaluation Criteria

In order to evaluate the performance of the models described in Section 4, the following metrics will be used.

- ***Accuracy:*** It is defined as the ratio of the number of correct predictions of the network across all the classes to the total number of input samples.
  As mentioned in the previous sections, accuracy works well only if an equal number of samples belong to each class (balanced data set).

$$Accuracy = \frac{TP + TF}{TP + TF + FP + FN} \tag{8}$$

- ***Precision:*** It is defined as the ratio of the number of correct positive predictions of the network to the total number of positive predictions.

$$Precision = \frac{TP}{TP + FP} \tag{9}$$

- ***Recall:*** It is defined as the ratio of the number of correct positive predictions of the network to the total number of all observations in the actual class. It is worth mentioning that there is no relation between precision and recall, and there is no guarantee that if a classifier has a high precision rate, it also has a high recall rate.

$$Recall = \frac{TP}{TP + FN} \tag{10}$$

- ***F1-Score:*** It is a metric that combines Precision and Recall metrics into a single metric. According to the F1-Score formula, it is the weighted average of the precision and recall.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN} \tag{11}$$

B. Confusion Matrix

It is a way to show a summary of the classifier's performance. The confusion matrix shows us the number of correct and incorrect predictions in each class. In this way, we can inform which class is more challenging for our classifier to predict correctly.

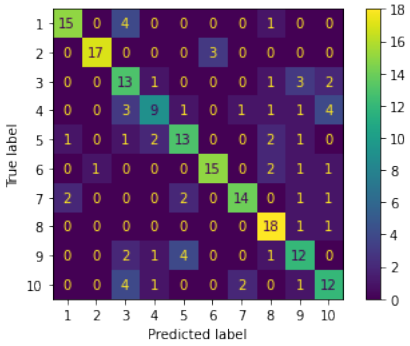

Fig. 9. Confusion Matrix of KNN Classifier

C. Quantitative Results

We presented the quantitative results on the tentative dataset in this section. As we said before, in order to comfort comparing the proposed classifier with other methods, we use four evaluation measures. Table I shows us quantitative results of each classifier. The evaluation score of each classifier is demonstrated. The green value denote the best result and the second best value is marked in red.
In our project the score of four different evaluation criteria is investigated and compare with each other, therefore in the Fig. 14. we illustrated the average score of 10 repeated 5-fold cross validation.

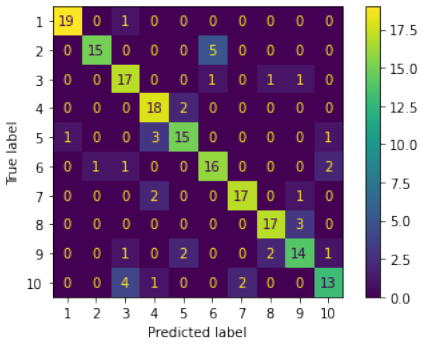

Fig. 10. Confusion Matrix of RF Classifier

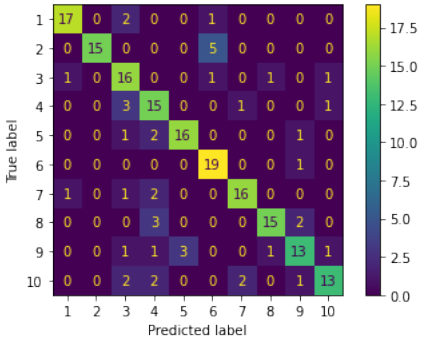

Fig. 11. Confusion Matrix of SVM Classifier

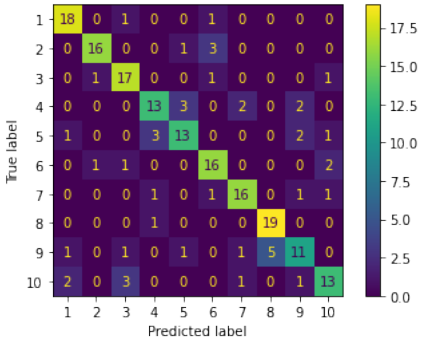

Fig. 12. Confusion Matrix of XGB Classifier

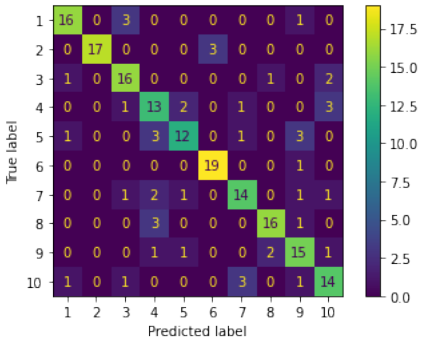

Fig. 13. Confusion Matrix of LR Classifier

TABLE I
THE RESULTS OF DIFFERENT CLASSIFIERS ON TESTING DATA

Classifier	Accuracy	F1-Score	Precision	Recall
Random Forest	80.5	80.47	81.08	80.5
Support Vector Machine	77.5	77.75	79.42	77.5
Logestic Regression	76	76.1	76.71	76
K Nearest Neighbors	69	69.16	70.64	69
Gradient Boosting	76	75.62	75.78	76

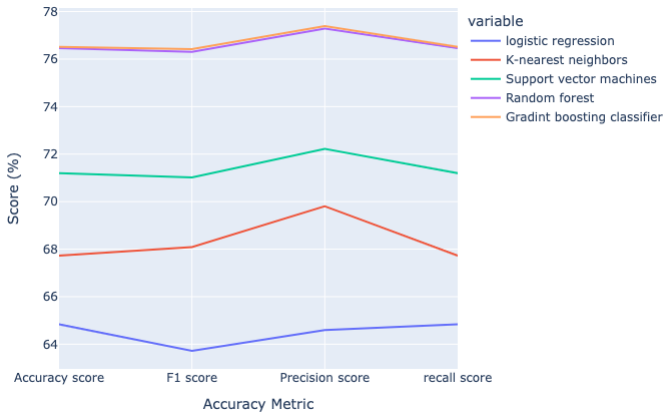

Fig. 14. Average score of 10 repeated 5-fold cross validation

V. CONCLUSION

Music genre classification is the process of using machine learning algorithms to classify music into different genres based on various characteristics. This can be useful for music recommendation systems, music streaming platforms, and other applications where it is important to understand the genre of a particular piece of music.

In this research work, we aimed to compare the performance of various classification algorithms for music genre classification problem. Five different methods were used including logistic regression (LR), Support Vector Machines (SVM), Random Forest (RF), K-Nearest Neighbors (KNN) and Gradient Boosting Decision Trees (XGB). We used the GAZN dataset, which contains 10 different music genres, each with 100 samples (Audio files). To evaluate the performance of different models, we implemented a repeated k-fold stratified cross-validation on training data. It should be noted that before cross validation stage, the hyperparameters of different models were tuned by grid search technique. We considered four different accuracy metrics including accuracy, f1, precision and recall score for the comparison of models. It was shown that RF and SVM had almost identical performance with different iterations and had higher accuracy than other models. Our results indicate that RF and SVM had the highest prediction accuracy (80.5%), while LR had the least prediction accuracy (76%) for unseen data.

REFERENCES

[1] M. Banitalebi-Dehkordi and A. Banitalebi-Dehkordi, "Music genre classification using spectral analysis and sparse representation of the signals," *Journal of Signal Processing Systems*, vol. 74, no. 2, pp. 273–280, 2014. 1

[2] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 2002. 1

[3] N. Pelchat and C. M. Gelowitz, "Neural network music genre classification," *Canadian Journal of Electrical and Computer Engineering*, vol. 43, no. 3, pp. 170–173, 2020. 2

[4] S. Vishnupriya and K. Meenakshi, "Automatic music genre classification using convolution neural network," in *2018 international conference on computer communication and informatics (ICCCI)*, pp. 1–4, IEEE, 2018. 2

[5] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 776–780, IEEE, 2017. 2

[6] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016. 2

[7] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Transactions on audio, speech, and language processing*, vol. 22, no. 10, pp. 1533–1545, 2014. 2

[8] L. Wyse, "Audio spectrogram representations for processing with convolutional neural networks," *arXiv preprint arXiv:1706.09559*, 2017. 2

[9] T. Li, A. B. Chan, and A. Chun, "Automatic musical pattern feature extraction using convolutional neural network," *Genre*, vol. 10, no. 2010, p. 1x1, 2010. 2

[10] T. Lidy, A. Schindler, *et al.*, "Parallel convolutional neural networks for music genre and mood classification," *MIREX2016*, vol. 3, 2016. 2

[11] J. W. Picone, "Signal modeling techniques in speech recognition," *Proceedings of the IEEE*, vol. 81, no. 9, pp. 1215–1247, 1993. 2, 3

[12] J. R. Deller, J. G. Proakis, and J. H. L. Hansen, "Discrete-time processing of speech signals," 1993. 3

[13] J. Benesty, "Springer handbook of speech processing," in *Springer Handbooks*, 2007. 3

[14] S. S. Stevens, J. Volkmann, and E. B. Newman, "A scale for the measurement of the psychological magnitude pitch," *The journal of the acoustical society of america*, vol. 8, no. 3, pp. 185–190, 1937. 3

[15] "https://link.springer.com/content/pdf/bbm:978-1-4614-6360-3/1.pdf," 3

[16] F. Zheng, G. Zhang, and Z. Song, "Comparison of different implementations of mfcc," *Journal of Computer science and Technology*, vol. 16, no. 6, pp. 582–589, 2001. 3

[17] T. Ganchev, N. Fakotakis, and G. Kokkinakis, "Comparative evaluation of various mfcc implementations on the speaker verification task," in *Proceedings of the SPECOM*, vol. 1, pp. 191–194, 2005. 3

[18] L. Rabiner and B. Juang, *Fundamentals of Speech Recognition*. Pearson Education signal processing series, Pearson Education, 1993. 3

[19] S. Furui, "Comparison of speaker recognition methods using statistical features and dynamic features," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 29, no. 3, pp. 342–350, 1981. 3

[20] J. S. Mason and X. Zhang, "Velocity and acceleration features in speaker recognition," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, pp. 3673–3674, IEEE Computer Society, 1991. 3

[21] Y. Amit and D. Geman, "Shape quantization and recognition with randomized trees," *Neural computation*, vol. 9, no. 7, pp. 1545–1588, 1997. 5

[22] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995. 5

[23] L. Jiang, Z. Cai, D. Wang, and S. Jiang, "Survey of improving k-nearest-neighbor for classification," in *Fourth international conference on fuzzy systems and knowledge discovery (FSKD 2007)*, vol. 1, pp. 679–683, IEEE, 2007. 6

[24] C. Bentéjac, A. Csörgő, and G. Martínez-Muñoz, "A comparative analysis of gradient boosting algorithms," *Artificial Intelligence Review*, vol. 54, no. 3, pp. 1937–1967, 2021. 6

[25] Y. Freund, R. Schapire, and N. Abe, "A short introduction to boosting," *Journal-Japanese Society For Artificial Intelligence*, vol. 14, no. 771-780, p. 1612, 1999. 6

[26] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001. 6