Music Genre Classification Using Various Classification Models

Priyanshu Kumar
Electrical Engineering
IIT Roorkee
22115123

Pukhraj Choudhary
Electrical Engineering
IIT Roorkee
22115124

Rawal Pritam B.
Electrical Engineering
IIT Roorkee
22115120

Abstract

This project focuses on the classification of music genres using machine learning techniques. We utilize pre-extracted audio features from a dataset of music tracks representing different genres. Various models, including K-Nearest Neighbours (KNN), Support Vector Machine (SVM), Random Forest, Decision tree, Gaussian Naive Bayes, and a Neural Network, are implemented and compared. Audio feature extraction, an essential aspect of audio signal processing, is explored using the Librosa library. The models are evaluated based on accuracy and performance metrics, with the neural network model achieving the highest accuracy. This project demonstrates the effectiveness of machine learning in analysing and categorizing complex audio signals into distinct genres.

I. Main Objectives

- To develop a system that can automatically classify music tracks into their respective genres using machine learning and deep learning techniques.
- To evaluate and compare the performance of various classification models, including traditional machine learning algorithms (like SVM, Random Forest, KNN) and neural networks for the task of genre classification.
- To optimize model performance by tuning hyperparameters, performing data preprocessing, and employing techniques like cross-validation and normalization.
- To analyse the strengths and weaknesses of different models and understand the challenges associated with music genre classification.

II. Status and Other details

- Completed
- Total Time spent – 3 weeks
- Percentage Contribution of each member
  - Priyanshu 33%
  - Pukhraj 33%
  - Pritam 33%

III. Introduction

Music is a universal language composed of complex signals that vary across genres, cultures, and time. Automatically identifying the genre of a music track is crucial for music streaming services, recommendation engines, and music information retrieval systems. Traditional methods relied heavily on human input and metadata, but machine learning has enabled automated genre classification by learning patterns directly from audio data.

This project aims to build and compare various machine learning models that classify music tracks into predefined genres based on audio signal features. We utilize pre-extracted numerical features from audio files and apply a suite of machine learning algorithms to predict the genre of each clip. The core of this task lies in understanding and leveraging the characteristics of audio signals.

IV. Understanding Audio Signal

Visualization of audio refers to the process of converting sound (audio signals) into a visual form, allowing people to see aspects of the sound. This can be done for analysis, music enhancement, education, or artistic expression.

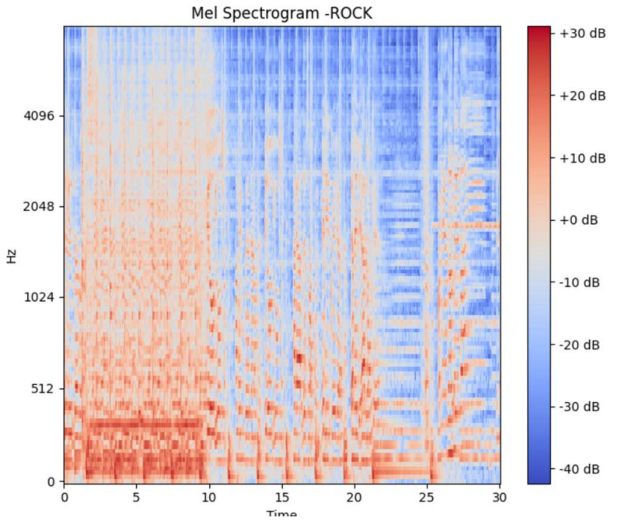
Mel Spectrogram Visualization:

A Mel spectrogram is a type of audio visualization that represents the time-frequency characteristics of an audio

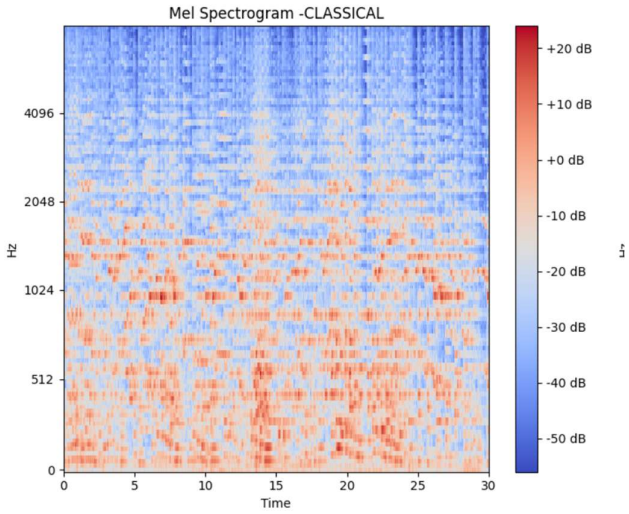
signal using the Mel scale, which is designed to approximate the way humans perceive sound. Unlike a traditional spectrogram that uses a linear frequency scale, the Mel spectrogram applies a perceptually motivated scale that spaces frequencies in a way that reflects how we hear pitch—more resolution in lower frequencies and less in higher ones.

Mel spectrograms are widely used in music genre classification and other audio-related machine learning tasks because they provide a compact and informative representation of the audio content. Since different genres often have distinctive rhythmic, harmonic, and timbral patterns, these characteristics can be captured effectively in the time-frequency domain and used as input features for classification models).

Why Do We Use the Fourier Transform in Audio Analysis? The Fourier Transform is used in audio analysis to convert signals from the time domain to the frequency domain. While time-domain signals show how sound changes over time, they do not reveal which frequencies are present. The Fourier Transform breaks the signal into its frequency components, helping us understand the pitch and tone characteristics. This is essential for tasks like music genre classification, where different genres have unique frequency patterns. It also forms the basis for creating spectrograms and Mel spectrograms, which are widely used in audio processing and machine learning to extract meaningful features from sound.



An audio signal is a representation of sound, usually in the form of a time-varying waveform. It can be analog

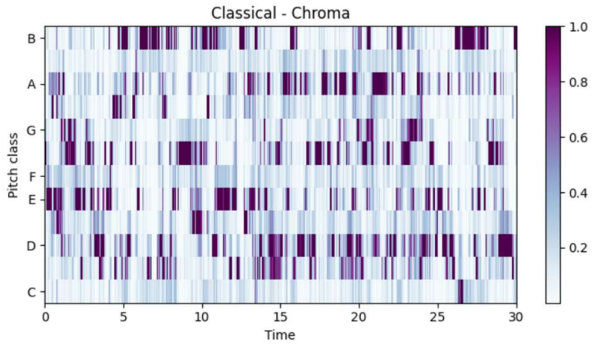


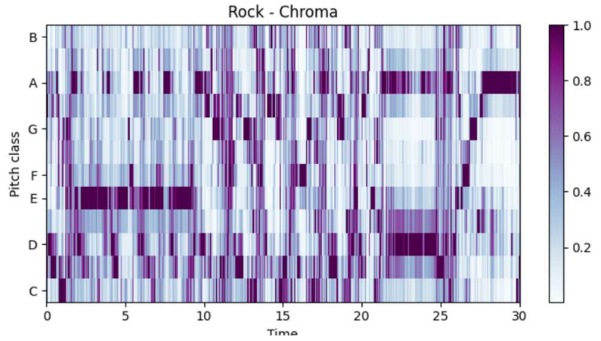
(continuous) or digital (discrete). In digital signal processing, audio signals are sampled at regular time intervals and quantized into numerical values, enabling computers to analyse and process them.

Several key concepts define audio signalling. The sampling rate refers to how many samples are taken per second (e.g., 44.1 kHz), which affects the quality and detail of the digital sound. Amplitude represents the loudness of the signal at a particular moment, while frequency determines the pitch of the sound, measured in cycles per second (Hz).

In this project, audio features are extracted using the Librosa library. These include:

MFCCs (Mel-Frequency Cepstral Coefficients): Capture the timbral texture of the audio and are commonly used in speech and music recognition.
Spectral Centroid: Reflects the "brightness" of a sound by indicating where the center of mass of the spectrum is located.
Chroma Features: Represent the distribution of pitch classes (such as C, D, E) in the audio.





Zero-Crossing Rate: Measures the rate at which the signal changes sign, helpful in detecting percussive sounds.

Tempo and Spectral Roll-off: Provide information about the rhythm and how quickly energy drops off in the higher frequencies.

These features convert raw audio signals into numerical form, making them suitable inputs for machine learning models used in music genre classification.

V. Theory

a) Random Forest

Random Forest is a popular machine learning algorithm works by creating a collection (or "forest") of decision trees during training time and then combining their results to make more accurate and stable predictions.

Each decision tree in the forest is trained on a random subset of the data and features. This randomness helps prevent overfitting and improves generalization. When making a prediction, each tree gives a result (e.g., a genre label for a music file), and the final prediction is made by taking the majority vote (in classification) or the average (in regression) of all the trees.

Random Forest builds multiple decision trees T_1, T_2, ..., T_n and combines their outputs to improve predictive performance. It uses bagging (bootstrap aggregation) and feature randomness to ensure diversity among the trees.

Each tree T_i is trained on a random subset of the training data

D is the full $D_i \subset D$, dataset. The training subsets are drawn with replacement, ensuring variability.

Let:

- x be an input feature vector (e.g., extracted audio features),
- $h_i(x)$ be the prediction of the i-th decision tree.

For classification, the final prediction $H(x)$ is made by majority voting among all the trees:

$$H(x) = \text{mode}\{h_1(x), h_2(x), ..., h_n(x)\}$$

For regression, the final prediction is the average of all tree outputs:

$$H(x) = \frac{1}{n}\sum_{i=1}^{n} h_i(x)$$

b) SVC

Support Vector Classification (SVC) is a supervised machine learning algorithm based on the concept of Support Vector Machines (SVM). It is primarily used for binary classification but can be extended to multi-class problems using strategies like one-vs-one or one-vs-rest.

The main idea of SVC is to find the best possible hyperplane that separates data points of different classes with the maximum margin. The margin is the distance between the hyperplane and the closest data points from each class, known as support vectors.

Given a training dataset of n points:

$$\{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}, \quad x_i \in \mathbb{R}^d, \quad y_i \in \{-1, +1\}$$

SVC aims to find a hyperplane defined by:

$$w \cdot x + b = 0$$

such that it maximizes the margin while correctly classifying the training data.

The optimization problem is:

$$\min_{w,b} \frac{1}{2}\|w\|^2$$

Subject to :

$$y_i(w \cdot x_i + b) \geq 1 \quad \forall i$$

To handle non-linear classification, SVC uses kernel functions $K(x_i, x_j)$ to map data into a higher-dimensional space where a linear separator is possible. Common kernels include:

- **Linear kernel:** $K(x_i, x_j) = x_i \cdot x_j$
- **Polynomial kernel:** $K(x_i, x_j) = (x_i \cdot x_j + 1)^d$
- **RBF (Gaussian) kernel:** $K(x_i, x_j) = \exp(-\gamma\|x_i - x_j\|^2)$

In this project, SVC was used to classify music into genres based on features extracted from the audio data. The algorithm was chosen for its ability to find optimal class boundaries even in high-dimensional and non-linear feature spaces, making it suitable for complex patterns found in music data.

c) Decision Tree Classifier

A Decision Tree Classifier is a supervised learning algorithm used for classification tasks. It works by splitting the dataset into smaller subsets based on feature values, creating a tree-like model of decisions. Each internal node represents a test on a feature, each branch represents an outcome of the test, and each leaf node represents a class label (e.g., a music genre).

The tree is built by selecting the features that best separate the data at each step. This selection is made using impurity measures, which determine how well a feature splits the data.
Common Impurity Measures:

1. **Gini Impurity**:

$$Gini(D) = 1 - \sum_{i=1}^{C} p_i^2$$

2. **Entropy (Information Gain)**:

$$P(C_k \mid x) = \frac{P(x \mid C_k) \cdot P(C_k)}{P(x)}$$

The feature that gives the highest Information Gain is selected to split the data:

$$Information\ Gain = Entropy(parent) - \sum_{k} \frac{|D_k|}{|D|} \cdot Entropy(D_k)$$

In this project, the Decision Tree Classifier was used to classify music into different genres based on features extracted using digital signal processing and autoencoders. The decision tree model is interpretable and provides insight into which features are most important for classification.

d) Gaussian Naive Bayes

Gaussian Naive Bayes is a probabilistic classifier based on Bayes' Theorem, with the naive assumption that features are independent of each other given the class label. It is especially effective and fast for classification tasks, even with high-dimensional data, and works well when the data features follow a normal (Gaussian) distribution.
In practice, we compute:

$$\hat{y} = \arg\max_{C_k} \; P(C_k) \prod_{i=1}^{n} P(x_i \mid C_k)$$

For continuous features, Gaussian NB assumes that the likelihood of the features is normally distributed:

$$P(x_i \mid C_k) = \frac{1}{\sqrt{2\pi\sigma_{C_k,i}^2}} \exp\left(-\frac{(x_i - \mu_{C_k,i})^2}{2\sigma_{C_k,i}^2}\right)$$

e) K-Nearest Neighbors Classifier

The K-Nearest Neighbors (KNN) algorithm is a simple, intuitive, and non-parametric method used for classification (and regression). It classifies a new data point based on the majority class of its K nearest neighbors in the feature space.
How It Works:

- Choose the number of neighbours K.
- Compute the distance between the new data point and all points in the training set.
- Select the K closest points (neighbours).

Assign the most common class among these neighbours as the predicted label.
Distance Metric:
The most commonly used distance metric is the Euclidean distance or Manhattan distance.
Prediction Rule:

$$\hat{y} = \text{mode}\left(y_{(1)}, y_{(2)}, \ldots, y_{(K)}\right)$$

$$Entropy(D) = -\sum_{i=1}^{C} p_i \log_2(p_i)$$

VI. Methodology

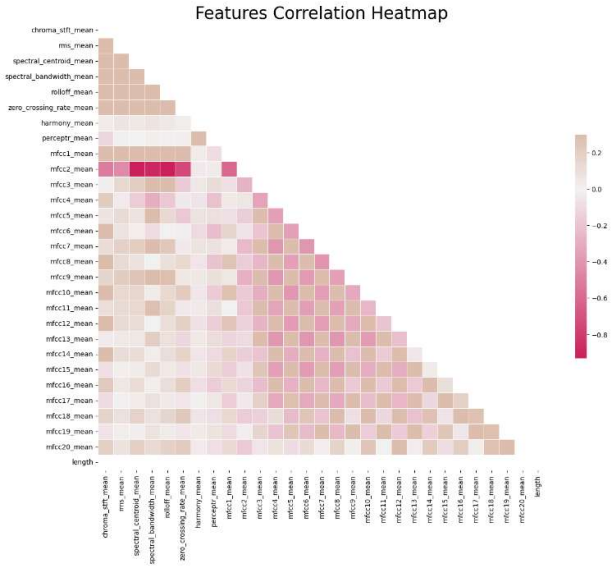
6.1 Dataset Description

The dataset used in this project is the GTZAN Genre Collection, a widely acknowledged benchmark in the field of music genre classification. It was compiled by George Tzanetakis and has been extensively used for research in content-based music information retrieval. The dataset comprises a total of 1,000 audio tracks evenly distributed across 10 different genres: Blues, Classical, Country, Disco, Hip-hop, Jazz, Metal, Pop, Reggae, and Rock. The dataset includes pre-extracted features for each audio file, stored in a CSV format. It contain extracted features from 9900 audio clips, and each clip has a fixed duration of 3 seconds.

6.2 Data preprocessing

The data preprocessing steps were carefully designed to ensure meaningful training for both models:

- **Clean Data:** Check for and handle any missing values in the dataset. The dataset was clean with no missing values. Removed irrelevant features like file name and length.
- **Analyse Features:** Compute aggregate statistics and visualize correlations among features to understand their relationships and assess the need for dimensionality reduction.



Correlation analysis between the extracted audio features indicated low multicollinearity, suggesting that feature reduction was not a necessary step and that the features are relatively independent and contribute unique information.

- **Encode Labels:** Convert categorical genre names into numerical labels using LabelEncoder.
- **Split Data:** Divide the data into 80% for training and 20% for testing. From this training data 20% of data is kept reserved for validation task later during model training.
- **Standardize Features:** Apply StandardScaler to standardize features for both training and test sets.

6.3 Model training and selection

Both traditional machine learning models and a deep neural network were explored for the classification task.

6.3.1 Traditional Machine Learning Models

Several classification algorithms were evaluated:

- Random Forest Classifier (RandomForestClassifier)
- Support Vector Classifier (SVC)
- Decision Tree Classifier (DecisionTreeClassifier)
- Stochastic Gradient Descent Classifier (SGDClassifier)
- Gaussian Naive Bayes (GaussianNB)
- K-Nearest Neighbors Classifier (KNeighborsClassifier)

Each model was trained on the standardized training data and evaluated on the standardized test data using "model_evaluation" function, which provides the test set score and a classification report. Based on the initial evaluations, the K-Nearest Neighbors Classifier showed the highest accuracy as shown in next section.

6.3.2 Hyperparameter Tuning for KNN

To enhance the performance of the chosen KNN model, a grid search was conducted using "sklearn.model_selection.GridSearchCV". The search explored a parameter grid including:

```
# Grid Search for best KNN parameters
param_grid = {
    'n_neighbors': [3, 5, 7, 9, 11],
    'weights': ['uniform', 'distance'],
    'metric': ['euclidean', 'manhattan']
}
```

Cross-validation (cv=5) was used with 'accuracy' as the scoring metric. The grid.best_estimator_ was selected as the optimal KNN model (model_KNN). Its performance was further analyzed on the test set using a classification report and a visualized with ConfusionMatrixDisplay.

The best parameters found are as follow:

```
best_estimator_: KNeighborsClassifier
KNeighborsClassifier(metric='manhattan', n_neighbors=3, weights='distance')
```

6.3.3 Deep Neural Network

A feedforward deep neural network was constructed using TensorFlow/Keras.

Architecture: The model is tf.keras.models.Sequential with: Several dense layers with decreasing units: 512, 256, 128, 64, 32 and 'relu' activation function. Each dense layer followed by a dropout layer (0.2) to reduce overfitting. Final layer with 10 units and softmax activation for multi-class classification.

Layer (type)	Output Shape	Param #
flatten_6 (Flatten)	(None, 57)	0
dropout_42 (Dropout)	(None, 57)	0
dense_48 (Dense)	(None, 512)	29,696
dropout_43 (Dropout)	(None, 512)	0
dense_49 (Dense)	(None, 256)	131,328
dropout_44 (Dropout)	(None, 256)	0
dense_50 (Dense)	(None, 128)	32,896
dropout_45 (Dropout)	(None, 128)	0
dense_51 (Dense)	(None, 64)	8,256
dropout_46 (Dropout)	(None, 64)	0
dense_52 (Dense)	(None, 32)	2,080
dropout_47 (Dropout)	(None, 32)	0
dense_53 (Dense)	(None, 10)	330

Total params: 204,586 (799.16 KB)
Trainable params: 204,586 (799.16 KB)
Non-trainable params: 0 (0.00 B)

Compilation: The model was compiled with the adam optimizer, sparse categorical crossentropy as loss function, and 'accuracy' as the metric.

Training: The model was trained on X_train and y_train for up to 1000 epochs (epochs=1000) with a batch size of 128 (batch_size=128). A validation split of 0.2 (validation_split=0.2) was used. An EarlyStopping callback with monitor='val_loss' and patience=15 was implemented to

halt training when the validation loss did not improve for 15 consecutive epochs.

VII. Results and Evaluation

Below we have tabulated the results for traditional ML classifiers :

Model	Train Accuracy	Test Accuracy	Precision	Recall	F1-score
RFC	**86**	**85.6**	**85.8**	**85.7**	**85.6**
SVC	**85.8**	**85.2**	**85.3**	**85.3**	**85.2**
DCT	**65.3**	**65.3**	**65.6**	**65.6**	**65.5**
NB	**50**	**50.7**	**51.6**	**51.1**	**48.9**
KNN	**89**	**89.4**	**89.6**	**89.5**	**89.4**

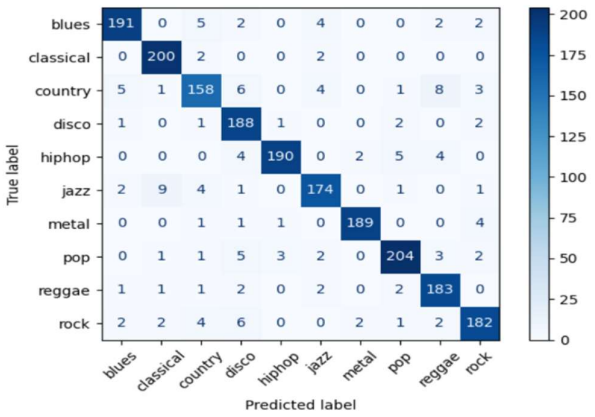
As we can see, KNN Classifier has the highest accuracy among other classifier models. Therefore, we will choose this model as our optimum model. We further fine-tuned it using Grid Search CV to identify the best hyperparameters and enhance its performance.

This table presents the classification report of the K-Nearest Neighbors (KNN) model.

Test set score: 0.931

	precision	recall	f1-score	support
blues	0.97	0.91	0.94	201
classical	0.90	0.99	0.94	180
country	0.90	0.87	0.88	208
disco	0.92	0.93	0.93	216
hiphop	0.97	0.91	0.94	186
jazz	0.94	0.89	0.92	214
metal	0.96	0.97	0.96	198
pop	0.96	0.96	0.96	203
reggae	0.91	0.98	0.94	198
rock	0.90	0.90	0.90	194
accuracy			0.93	1998
macro avg	0.93	0.93	0.93	1998
weighted avg	0.93	0.93	0.93	1998

This figure illustrates the confusion matrix for the K-Nearest Neighbours (KNN) model:



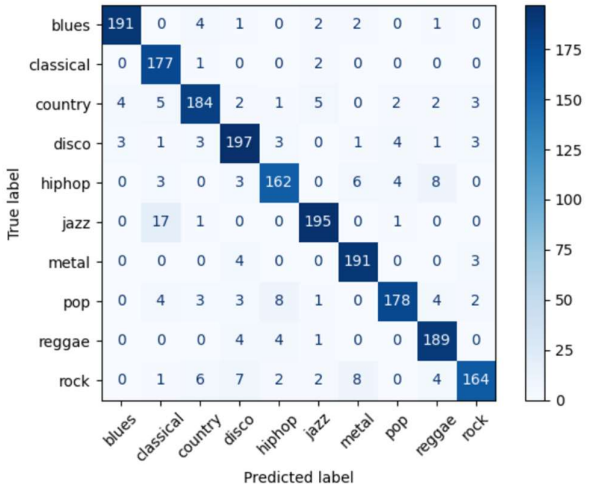
Using Neural Networks

This figure presents the classification report of the Deep neural network :

Test set score: 0.915

	precision	recall	f1-score	support
blues	0.96	0.95	0.96	201
classical	0.85	0.98	0.91	180
country	0.91	0.88	0.90	208
disco	0.89	0.91	0.90	216
hiphop	0.90	0.87	0.89	186
jazz	0.94	0.91	0.92	214
metal	0.92	0.96	0.94	198
pop	0.94	0.88	0.91	203
reggae	0.90	0.95	0.93	198
rock	0.94	0.85	0.89	194
accuracy			0.91	1998
macro avg	0.92	0.92	0.91	1998
weighted avg	0.92	0.91	0.91	1998

The confusion matrix for the Neural Networks applied to the task of music genre classification :

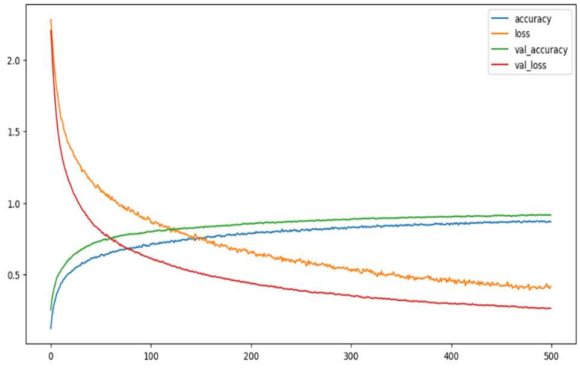


Below figure displays the training and validation accuracy and loss curves for a deep learning model trained over 500 epochs. The metrics shown include:

- Accuracy (blue) and Loss (orange) for the training set.
- Validation Accuracy (green) and Validation Loss (red) for the validation set.
- The plot illustrates a consistent improvement in model performance during training:
- Training loss and validation loss both decrease steadily, indicating the model is learning effectively and generalizing well.
- Training accuracy and validation accuracy increase over time, with validation accuracy closely tracking the training accuracy, suggesting minimal overfitting.

By the end of training, both accuracy curves plateau while the loss continues to decrease slightly, showing convergence. The

absence of divergence between validation and training performance indicates that the model is well-regularized and performs reliably on unseen data.



VIII. Observations

- The dataset was clean with no missing values, and feature correlation was low, indicating no need for feature reduction.
- Visualizing audio features like waveforms and spectrograms showed clear differences between genres, supporting the use of these features for classification.
- Among the initial machine learning models tested, K-Nearest Neighbours (KNN) showed the best performance.
- Hyperparameter tuning improved the KNN model's accuracy.
- Neural Network achieved the a second highest accuracy overall.
- Neural Network generalized better on external songs.
- The test on a Bollywood song illustrate how each model processes out-of-sample data, revealing classification challenges likely due to the dataset's reliance on 20-year-old Western music..=

IX. Conclusion

Based on the evaluation of the models, among traditional models, KNN gave the best performance with test accuracy of 93.1%, while the Neural Network achieved the second highest overall accuracy of 91.5% on test data. However, neural network gave better result with out of sample data. This demonstrates its effectiveness in distinguishing between different music genres using the extracted audio features.

X. Future Work

- Use raw audio inputs and apply CNNs on spectrogram or waveforms for deep learning.
- Explore LSTM and Transformer-based models, which process audio as time-series data to capture temporal dependencies.
- Modern music often blends multiple genres, like blues with classical or indie pop with metal and jazz. Incorporating fusion genres can improve the model's performance and commercial value.
- The GTZAN dataset and its genres focus mainly on Western music, but there are many other styles out there, like Indian, Asian, and Middle Eastern music, that aren't covered.
- Applying audio data augmentation techniques such as time stretching, pitch shifting, and noise addition to improve model generalization.
- Incorporate metadata such as artist or release year can help the model make more informed predictions.

XI. References

[1]. Survey Of Music Information Needs, Uses, And Seeking Behaviours: Preliminary Findings by Jin Ha Lee, J. Stephen Downie published in 2004 in ISMIR.

[2]. "Visualising music: the problems with genre classification" by Janice Wong.

[3]. Automatic Musical Genre Classification Of Audio Signals by George Tzanetakis, Georg Essl and Perry Cook. Department of Computer Science, Princeton University.

[4]. "Music type classification by spectral contrast feature." by Jiang, Dan-Ning, Lie Lu, Hong-Jiang Zhang, Jian-Hua Tao, and Lian-Hong Cai.

[5]. "Detecting Harmonic Change In Musical Audio" by Christopher Harte and Mark Sandler, Centre for Digital Music, Queen Mary, University of London, UK and Martin Gasser, Austrian Research Institute for Artificial Intelligence (OFAI), Vienna, Austria.

[6]. "Music Genre Recognition" by Karin Kosina.

[7]. "Single-layer learning revisited: A stepwise procedure for building and training a neural network" by Knerr, S., Personnaz, L., and Dreyfus.

[8]. "Support Vector Machines for Pattern Classification" by Shigeo Abe.

[9]. Music Genre Classification by Archit Rathore, Margaux Dorido from Indian Institute of Technology, Kanpur.