



Building a Classification and Recommendation Model for Music Genres

(A machine learning approach)

2023A2PS0296P	Rudra Narayan Bhatt
2023A2PS0235P	Smita Prajna Prusty
2023A8PS0709P	Arpit Kumar Khatri
2023A2PS0232P	Agrim Gupta
2023A2PS1301P	Pinak Sharma

Topic Number: 25

Group Number: 8

Progress (till 21/10/2025)

→ Completed the **E.D.A. (exploratory data analysis)** on the data, explored the waveforms(fig:3) and spectrograms(fig:2) for various genres to get a rough estimate of observable patterns. We got an idea about the entirety of the dataset. We observed that there are indeed no missing values or class imbalance in the GTZAN Kaggle dataset. From the figure below, we can see that there are 100 samples from each of the 10 classes(fig:1). However, we do introduce a certain degree of imbalance while creating the train-test split.

label	
blues	100
classical	100
country	100
disco	100
hiphop	100
jazz	100
metal	100
pop	100
reggae	100
rock	100

Fig : 1

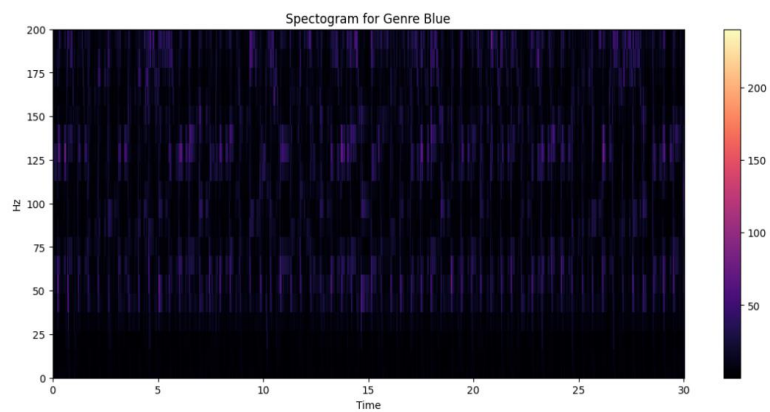


Fig : 2

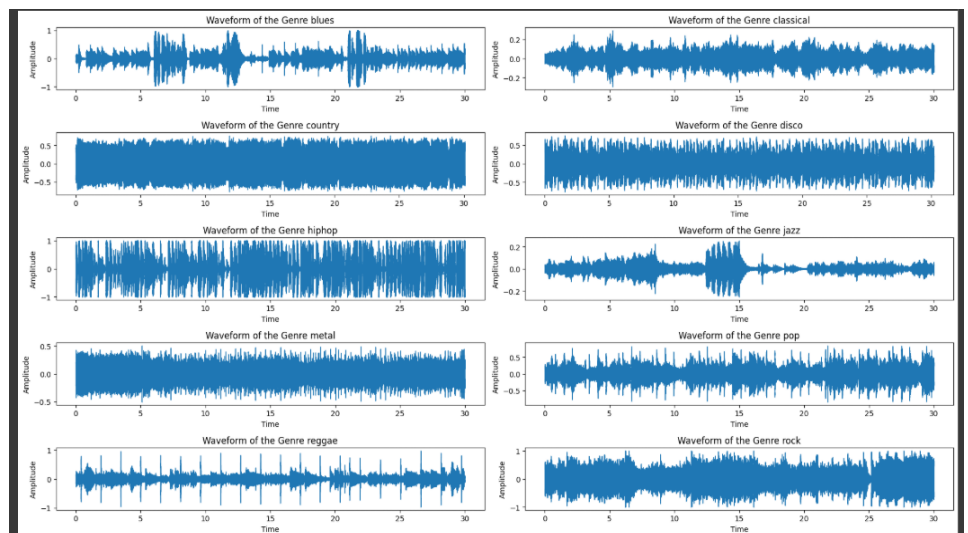


Fig : 3

→ In addition to observing the waveforms and spectrograms for the genres, we also calculated the **feature correlation heatmap** (fig :4) as a part of EDA, wherein we visualized the degree of correlation between each of the MFCC (mel frequency cepstral coefficients) coefficient for the entire dataset to help detect redundancy or understand feature-relationships.

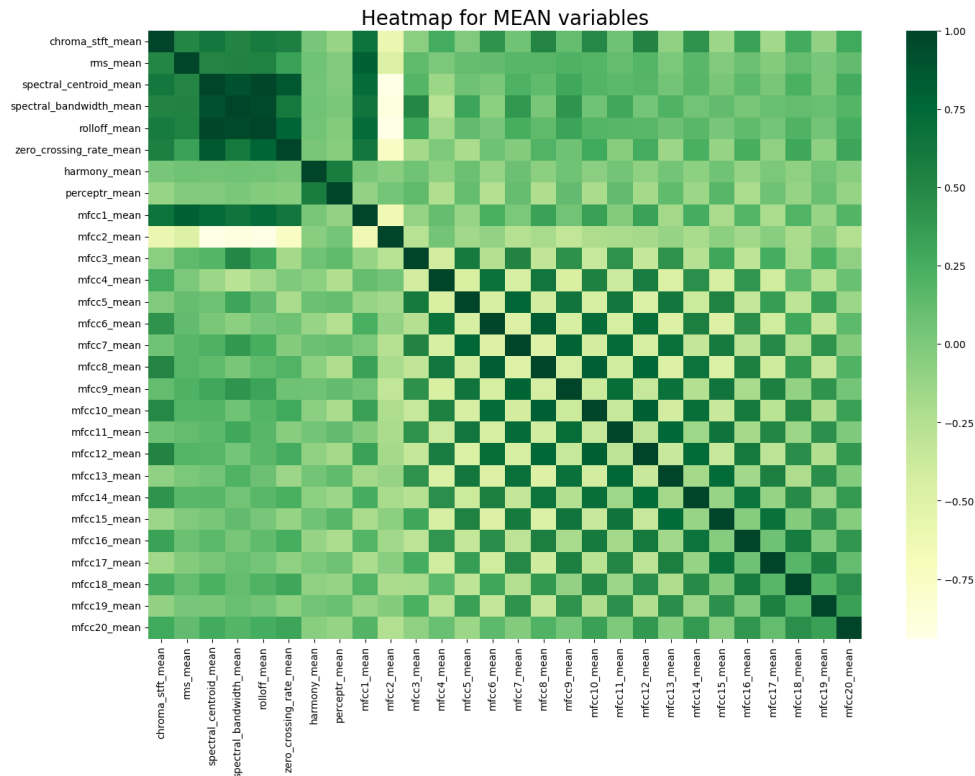


Fig: 4

→ Having completed the EDA, we did the preprocessing for the baseline (K-nearest neighbours) and the intermediate (Random Forest) models separately.

(i) K-Nearest Neighbors (K.N.N.):

So instead of using the regular/inbuilt KNN functions, we built a custom algorithm and therefore we required a different preprocessing algorithm. Instead of using any of the available csv files (for 30 second and 3 second time windows), we tracked the MFCC variation for every 20ms time frame for each .wav file and at the end filtered out 3 essential fields for our classification model:

- Mean_feature matrix – which stores the mean of the MFCCs across all time frames.
- Covariance matrix – which stores the covariance across all time frames for each MFCC.
- Label – which stores the class labels for each data point.

We did this instead of choosing any of the available csv files because we wanted to have as much control over the feature engineering of the data as possible (note that we also tried using the available csv files with the random forest model).

Since the feature-set that we considered was much different from the feature -set in the csv files, we had to consider **Mahalanobis distance / KL divergence** instead of Euclidean distance (which would make very little sense in this case since there are more than about 25 features).

Lastly, we considered the value of K as 5 (though for larger K, we could have attained a higher accuracy). However, the modified KNN approach did produce a considerably high accuracy of about **70%** for such a simple model.

(ii) Random Forest:

Random forest is a bagging ensemble method where each base-classifier is a decision tree each trained on bootstrapped data based on 'bagging' principle (feature-bagging in our case).

The preprocessing for the Random Forest model was fairly simple. We used the 30 second time window csv files and scaled each feature using the '**miniMax**' algorithm instead of the '**standard scaling**' algorithm. We used this because standard scaling could produce negative values for certain coefficients which would not affect the model predictions but would make it less interpretable and make no sense.

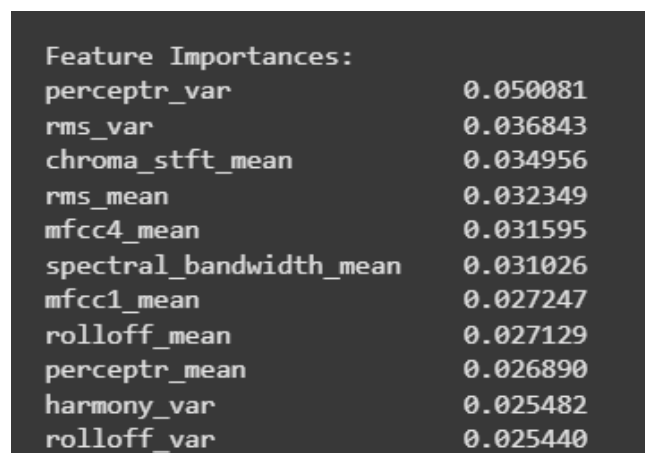
Finally, for the hyperparameters we tried training at different numbers of base classifiers and maximum tree-depths, however we got best results at number of base classifiers =200 and when each tree is fully grown. We arrived at a test accuracy of 88% which is impressive considering the minimum feature engineering.

➔ Comparison between Random Forest and KNN:

We did modify the regular KNN approach to attain higher accuracy (of about 70%) however, the regular KNN produced an accuracy of about 27% in contrast to the 88% of the random forest. This shows that the random forest ensemble model generalizes better in comparison to KNN which relies solely on distance calculations. An additional reason for better accuracy could be due to better noise-handling capability of random forest compared to KNN (because bagging distributes the effect of noise across all base classifiers and therefore its effect on the final decision reduces significantly).

Additionally, the initial training time for random forest may be high owing to fact that it has to train a number of base-classifiers independently in comparison to the zero-training time for KNN. However, the inference/prediction time for random forest (0.27 seconds) is much faster compared to KNN (10 seconds, though this is subject to the hyperparameter k).

Also, this may be noted that random forest analyses feature and can quantify their usefulness while KNN uses all features blindly and provides for no feature importance (Fig : 5 shows the top features by importance for the random forest model).



Feature Importances:	
perceptr_var	0.050081
rms_var	0.036843
chroma_stft_mean	0.034956
rms_mean	0.032349
mfcc4_mean	0.031595
spectral_bandwidth_mean	0.031026
mfcc1_mean	0.027247
rolloff_mean	0.027129
perceptr_mean	0.026890
harmony_var	0.025482
rolloff_var	0.025440

Fig : 5

List of Tasks Remaining:

→We also intend upon implementing Deep Learning approaches like Convolutional Neural Networks as a part of our advanced model analysis.

→We aim at implementing additional ML algorithms like SVM, gradient-boosting and a SoftMax regression approach at as proposed earlier in the proposal as a part of our innovation and stretch objectives.

→We also wish to build a full-scale music genre-recommender using **SVD** (singular value decomposition) or **ALS** (alternating least squares).

END