CSE 343: Machine Learning Final Report (Winter 2022)
Music Recommendation System

Jasdeep Singh 2019047
Siddharth Singh Kiryal 2019277
Shanu Verma 2019104

1. Introduction

In the present day scenario, most people listen to music. There are 79 million songs present on the internet. The quantity of songs has exploded due to the increased popularity of music streaming services. It has grown laborious and difficult for consumers to get hold of similar songs of their preference. It includes individuals listening to and classifying various tunes based on their acoustic capabilities. Moreover, the recommendation system based on the song's metadata may not give them the best user experience. We wish to provide them with a better music experience by audio sampling the wav files and extracting the features. The goal is to improve efficiency and introduce automation for the said task. Machine learning techniques of today and Visualization tools should aid in discovering accurate models which recommend songs for users on music selections, as this was not addressed in previous research.

2. Related Work

We have implemented Logistic Regression, KNN, ANN, Clustering, and CNN.

2.1 K-Nearest Neighbors (KNN)

We referred to [12]. There they implemented KNN with other models. They used the mean and variance of the genre to analyze data and note interesting observations.

2.2 Clustering

We went through paper reference [11] for the clustering algorithm, where they wrote their implementation of K-Mean. They chose to represent a centroid as if it were also a multivariate Gaussian distribution. They were able to achieve 80% accuracy.

2.3 Convolutional Neural Network (CNN)

In the CNN model applied in reference [10], the initial accuracy using three convolution layers was 75.65%. However, later on, optimization led to an improvement to 85.36%. Here the input is first traversed through convolutional layers, followed by RNN/GRU layer to classify.

3. DataSet and Evaluation

Obtained 1000 files from the GTZAN dataset (link)[1]. Added two genres with 100 wav files each (Electric and k-pop). Extracted 30 sec from each wav file (from 30s offset) for further audio signal processing. Extracted features include RMS, chroma_stft, spectral_centroid, and many others using Librosa to create a 1200x43 dimensional dataset. For testing ANN, KNN, and logistic performance, the dataset was divided into 720:480.

The spectrogram and wavelet plots of all the 30s audio files were generated, shuffled, and divided into train and test datasets for image-based classification purposes. Spectrogram plots were used as inputs for image-based genre classification using CNN modes. The images were rescaled and converted into 256 x 256 x 3 (RGB) format. Then all the pixel values for all the images were standardized using the formula:

$$I[i][j][k] = \frac{I[i][j][k] - mean(I)}{standard\ deviation(I)}$$

, where I is the image of a 3 dimensional array. This was followed by fitting several set image control parameters to the training dataset. All 12 genres were encoded (0-11) for coding and NN classification purposes. CNN model uses a 60:40 train-test split (720:480 samples).

In ANN, CNN, and KNN, the evaluation metrics use accuracy, precision, F1-score, and Recall score. Further insights into the model have been obtained by analyzing the confusion matrix, heat maps, and learning curves (for train-test accuracy and loss). Silhouette and Davies Bouldin Scores evaluate clusters' separation (purity) for clustering models.

4. Methodology

4.1 Content Recommendation approach

There are three main approaches to content recommendation.

4.1.1 Demographic Filtering

The first is Demographic Filtering, where recommendations are based not on the user preferences

but on the overall ranking of the songs based on global reviews received by the songs.

This technique is much better than randomly recommending songs and it is useful when you want to recommend content that is generally favored by the larger population of viewers. But of course, this approach does not at all factor in user preference.

4.1.2 Content-Based Filtering

The second approach is Content-Based Filtering. This technique uses the metadata about the song such as song title or artist name, track length, features, and genres of the songs to find other songs which are similar to the songs selected by the user. When the user selects a song, the song is matched against other songs in the database to find the songs that are the most similar to the selected song based on the above-mentioned parameters. The top matches are then presented to the user and this is how content-based filtering works in essence. The similarity metric being used can be something like cosine similarity, Pearson similarity, or any other such metric. This approach does take into account the user's selection however it may fail to capture the general taste and inclination of the user.

4.1.3 Collaborative Filtering

The last approach is that of collaborative filtering. This approach does not require the song metadata to make recommendations, instead, it relies on the user ratings made by other users of the service. This approach tries to match the general taste and behavior of the active user to existing users in the database and then tries to predict what the active user might like. A major downfall of this approach, however, is that data sparsity can negatively impact its performance. The data about user ratings and reviews are likely to be sparse. Thus, used on its own, the technique is prone to poor performance.

Meta features of songs such as spectral bandwidth, RMSE value, zero-crossing rate, frequency, harmonic mean and many others are used to make the dataset.

As our objective is to make a recommendation system. For that, we require to have a collection of models that are best suited for the dataset. Hence, our initial step would be to explore all the models and find out which models are giving better performance in terms of a better fit for both training and testing sets. After that, we would use the selected models to make a recommendation system along with recommendations metrics.

For the recommendation system, we would be using **content based filtering** for the recommendation.

4.2 Model Details

4.2.1 KNN and Logistic Regression

In KNN and logistic regression, without preprocessing, the accuracy was around 30-34% for train data and 50% for test data, which showed that the dataset is sparse. We need to reduce the sparsity to increase the model's performance. Normalization, Standardization, PCA, Truncated SVD, and SelectKBest were applied with parameter tuning, improving the model performance to 55-60%. Further grid search and hyperparameter tuning led to 66% and 73% accuracy in KNN and Logistic regression. Many strategies were then explored to improve performance; however, no difference was found due to the sparsity of the data. Based on learning curves evaluation on test and training data, we verified that the model is not overfitting and underfitting.

4.2.2 Random Forest

In the random forest, with a simple model (RandomForestClassifier(random_state=0)), we are getting around 100% accuracy on testing data and 66% accuracy on cross-validation data and 58% accuracy on testing with more than 20 'n_estimators' (Figure -18). It clearly shows that our model is overfitting the data. Then, we tried to apply parameter tuning using GridSearchCV (Parameters - n_estimators=275,min_samples_split=9, min_samples_leaf=3,max_depth=145,bootstrap=True). But our parameter tuned model performed the same on testing data(accuracy 58%) but our validation accuracy increased a little bit (Figure - 19).

4.2.3 Clustering

In Clustering, we applied K-Means, Birch, and Gaussian Mixture Model. We plotted a Silhouette score corresponding to each cluster (figure 3). We used cluster separation as a metric (measured in Silhouette and Davies Bouldin Score) instead of accuracy, as clustering will assign an arbitrary label to every cluster it forms. Therefore, comparing assigned labels with actual labels to evaluate accuracy will not be the best measure. To improve the cluster's purity (or separation), we also tuned different clustering hyperparameters (mentioned in the results). We tried PCA with a different number of n_components(figure 4) and noticed that PCA with n_component=2(Variance Explained: 64.65) gave us the best results. To find why lower n_components in PCA are performing better, we used TSNE (figure 6) and plotted our data points. Our data is very sparse; hence, clusters formed with higher n_components will not be that separated, resulting in a lower Silhouette Score.

4.2.4 Artificial Neural Network (ANN)

In ANN, this model has been applied to the metadata

(1200X43) dataset. The ANN architecture includes three Relu layers. The accuracy for the test set was coming out to be 72%. On analyzing the test set accuracy and train set accuracy, it was found that the model is overfitting the data. L1 & L2 regularization, Dropout (0.2), reduction in each layer, and batch normalization were applied as the overfitting counter. After modeling and choosing the correct parameters (hyperparameter tuning), we reduced the overfitting to a certain extent. The final ANN model consisted of 2 L1 regularizer layers and an output softmax layer. After each layer, batch normalization and Dropout (0.3) were added. The test accuracy was improved to 80.8% (Figures 1 & 2).

4.2.5 Convolutional Neural Network (CNN)

The CNN architecture (Figure8) includes three relu activated convolution layers, max-pooling followed by batch normalization. The first dense layer(64) has a dropout of 0.3. Initially, the accuracy was 27%. The model could not extract features efficiently due to a small loaded dataset insufficient for training. Accuracy stagnated at 34% on increasing dataset volume. A third convolution layer (to increase trainable parameters for more feature extraction) was added to increase the accuracy. Full-sized dataset(around 243MB) was used for the train-test split. The number of epochs was increased, and batch size was slightly decreased. Max pooling was used to reduce successive outputs size while reducing possible noise impacts and not affecting sharp recorded features in the convolution's input. This improved accuracy to 40%, but now the model was overfitting after around 52 epochs (Figure 9,10). To reduce the overfitting, standardization was applied, which made features of pixels more readily available/sensitive/distinguishable. This was followed by applying batch normalization to all convolution layers and applying a 30% dropout in the first dense layer, using batch_size=32, reducing the number of epochs from 180 to 50 and the learning rate from 0.001 to 0.0001. This improved accuracy to 52% (Figures 9). We used Adam optimizer and Sparse Categorical Cross-Entropy for loss. There is still some overfitting left, and to remove it, image enhancement, regularization, layer count reduction to 2, fine-tuning epoch number(around 52) and train-test(around 80:20) ratios, adding more images, and early stopping will be used.

4.2.6 Support Vector Machine (SVM)

The preprocessing is applied in similar fashion as done in 4.2.1. In initial stages the accuracy was observed to be around 68%. On applying GridSearchCV on the dataset we were able to increase the accuracy to 74. The best parameter was found to be SVC(C=10, gamma=0.01).

5. Results and Analysis

5.1 Supervised Learning

In Logistic regression, CNN, ANN, and SVM - Accuracy, recall, f1 score, precision and curve for loss or accuracy on training and testing data was used to find the best fit for the data. Preprocessing was done in logical order to bring data to its best possible form. Results can be seen in the table below.

Model	Accuracy	Precision	F1 score	Recall score
KNN	0.6667	0.6892	0.6693	0.6667
Logistic Regression	0.7333	0.7503	0.7376	0.7334
ANN	0.8083	0.7927	0.8042	0.8023
CNN	0.52	0.5175	0.5083	0.5166
Random Forest	0.58000	0.6783	0.665	0.685
SVM	0.73	0.74	0.73	0.72

Table 1: Accuracy, Precision, F1 and Precision scores for different models observed

5.1.1 Convolutional Neural Network (CNN)

Figures 9,10 show that the CNN model still shows some degree of overfitting starting from the 52 epoch (As accuracy is starting to stabilize, but the loss is again increasing). Prediction is accurate, most for classical and worst for reggae genres.
It can be said that if any genre A is being mistaken for genre B then this is because the spectrograms of genre A in the trained data set are more generally similar to those of genre B -> Genre A being the superset of genre B in similarity.

The training and validation losses and accuracies are found to settle while training (reaching 175 epochs), while the actual values are fluctuating quite a bit around the average. This is because of noise which is found in the image input for predictions, where many images are found to be similar.

There is a large difference observed between training and validation loss and accuracy because the model that is being trained quite well on the training dataset, is not able to give much good predictions on the testing dataset. This might be because of the small size of the training and testing dataset volume used/possible underfitting.

5.1.2 Artificial Neural Network (ANN)

In ANN, we can infer from train-test error and loss graphs that the model is not overfitting nor underfitting the dataset. This is because the training accuracy is neither too high nor too low. (Figure 7)

5.1.3 KNN, Logistic Regression, and SVM

Considering the analysis of state of art KNN [12], the model was made, which gave good performance results (69 %). In this model jazz and rock were the genres which were giving lower accuracy 57 and 62. Rest all genres were giving similar results.

In the logistic regression model, rock and disco are giving accuracy results - 57 and 59. And the best genre is k-pop

In SVM, the country and disco genre are giving poor results - 55 and 47. And the best genre in SVM is found to be Electric-Genre.

5.2 Unsupervised Learning

5.2.1 Clustering

In Clustering, we have used Silhouette Score[2] Davies Bouldin Score[3] to check clusters' separation (purity) instead of analyzing accuracy. (Table2). We used different techniques to increase the Silhouette score and decrease the Davies Bouldin score.

Model	**Hyperparameters**	**Silhouette Score**	**Davies Bouldin score**
K Means	n_clusters=9 n_init=9 algorithm='auto'	35%	83%
Birch	n_clusters = 9 threshold = 1.9 branching_factor=20	32%	68%
Gaussian Mixture	n_components = 9 covariance_type='spherical'	52%	66%

Table2: Hyperparameters, Silhouette Scores, and Davies Bouldin Scores for different clustering models

5.3 Recommendation

5.3.1 Clustering

Using clustering methods, we create a recommendation system. When a user selects a song, we take all of the songs in the same cluster as the selected song for all clustering models when making recommendations. In PCA projected space, we now calculate the euclidean distance between the picked song and the recommended song. We will choose the best 10 shortest distance songs from a list of songs for recommendation(Figure 17).

5.3.2 K-Nearest Neighbors (KNN)

Recommendation system was built using the best fitted model observed during classification. On that we applied the euclidean distance measure to observe the nearest datapoint in n dimensional space. In our case, we sorted distance on the basis of increasing distance and obtained the first 6 points. Then using this datapoint we predicted the music file to the user.

5.3.3 Artificial Neural Network (ANN)

The Recommendation System was built on the best fitted model. We picked the in-between layer, in our case we picked the layer('dense1') with the number of neurons - 128. The model was used to predict the output only till the 'dense1' layer. (Figure 20). The output is called the embedding. Cosine similarity of sklearn.metrics.pairwise library is used. Cosine similarity measures the similarity between vector lists by calculating the cosine angle between the two vector lists.

Then when the user inputs the file name present in the database, we find the feature vector corresponding to the file. After that this vector is processed by the neural network till the 'dense1' layer and the output vector is obtained. Then we calculate the similarity score for all the files and sort this in decreasing order of similarity score. The top 6 indexes are used to find the file name and recommend it to the user. The recommendation for the blues genre was found to be the most accurate.

After analyzing the recommendation from different models we found that recommendation results showed most suitable songs in case of ANN across all the genres. In ANN, we are able to find the music files which have features that are 88% similar to the recommended songs (Figure 14). This may be due to the ANN giving the best results on the training and testing data for loss and accuracy.

6. Contributions

Shanu Verma: Data Collection and preprocessing,Feature Extraction using Librosa, Data Dimension Reduction - PCA, TSNE, Normalization, KMeans, DBSCAN, Birch and Gaussian Clustering, Random Forest. Recommendation using Clustering algorithms

Jasdeep Singh: Data Scraping / Collection, Feature Extraction using librosa, Data analysis, Data Visualization, PCA, Normalization, Standardization, Truncated SVD, SelectKBest, Logistic and Analysis, SVM and Analysis, ANN and Analysis, GridSearchCV and Analysis. Recommendation system - ANN, KNN

Siddharth Singh Kiryal: Data Scraping/Collection and preprocessing: song length reduction (to 3s), spectrogram and waveform generation, feature extraction, spectrogram image train-test generation, Dimensionality Reduction- Normalization, Standardization, Image transformations, genre prediction, , analysis and evaluation - heatmaps plots, test, validation loss accuracies plots etc.

APPENDIX

References

[1] https://www.kaggle.com/andradaolteanu/gtzan-dataset-music-genre-classification
[2] https://towardsdatascience.com/silhouette-coefficient-validating-clustering-techniques-e976bb81d10c
[3] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.davies_bouldin_score.html
[4] https://towardsdatascience.com/silhouette-coefficient-validating-clustering-techniques-e976bb81d10c
[5] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.davies_bouldin_score.html
[6] https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/
[7] https://machinelearningmastery.com/overfitting-machine-learning-models/
[8] https://github.com/christianversloot/machine-learning-articles/blob/main/how-to-check-if-your-deep-learning-model-is-underfitting-or-overfitting.md
[9] https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/#:~:text=Overfitting%3A%20Good%20performance%20on%20the,poor%20generalization%20to%20other%20data
[10] https://www.researchgate.net/publication/237054335_The_GTZAN_dataset_Its_contents_its_faults_their_effects_on_evaluation_and_its_future_use
[11] https://cs229.stanford.edu/proj2011/HaggbladeHongKao-MusicGenreClassification.pdf
[12] https://www.researchgate.net/publication/334780384_Automatic_music_genre_recognition_for_in-car_infotainment#pf7

Figures

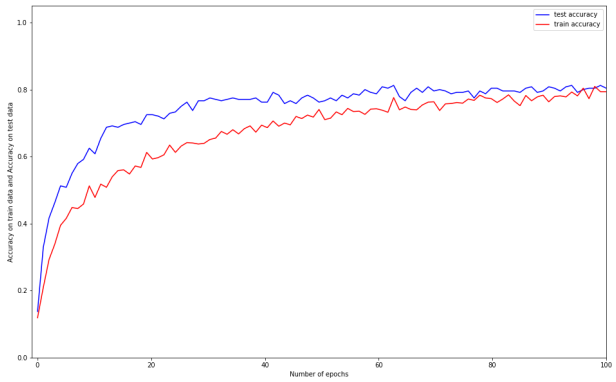

Figure 1: ANN - Varying test and train accuracy with epochs

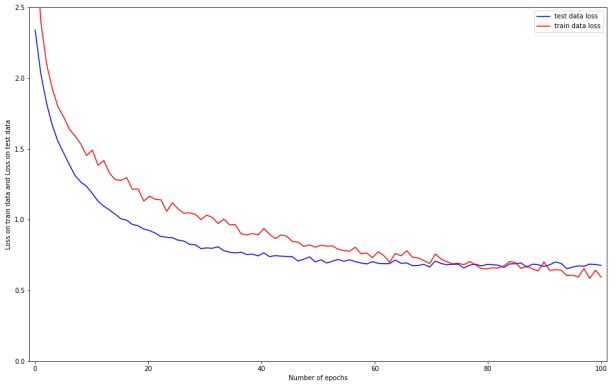

Figure 2: ANN - Varying test loss and train loss with epochs

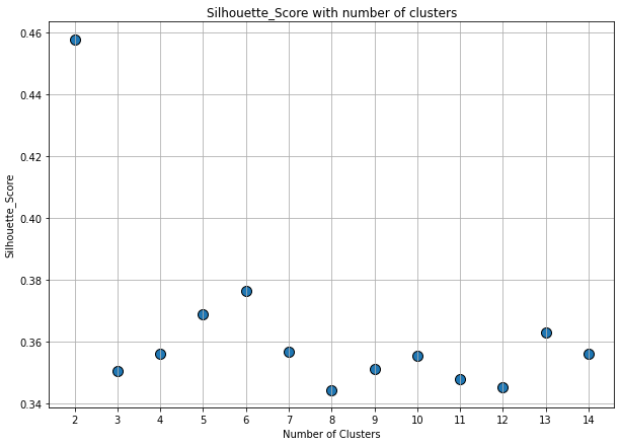

Figure 3: Silhouette_Score with number of clusters (K_Mean)

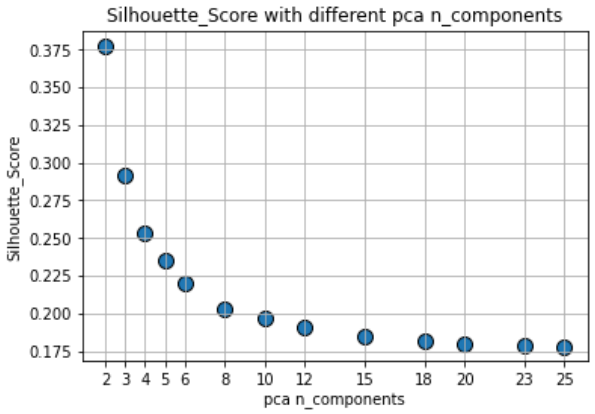

Figure 4: Silhouette_Score with different PCA components in K_Mean(n_clusters=9 n_init=9)

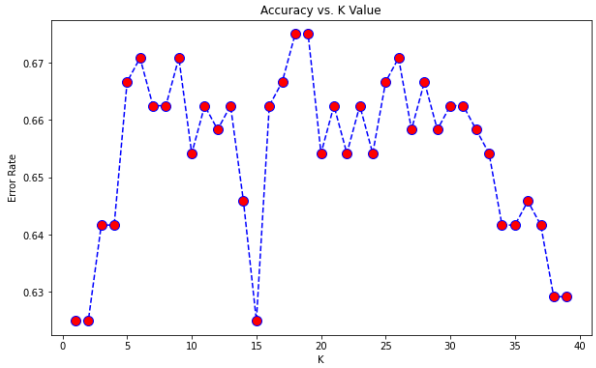

Figure 5: Accuracy varying with k number of neighbors

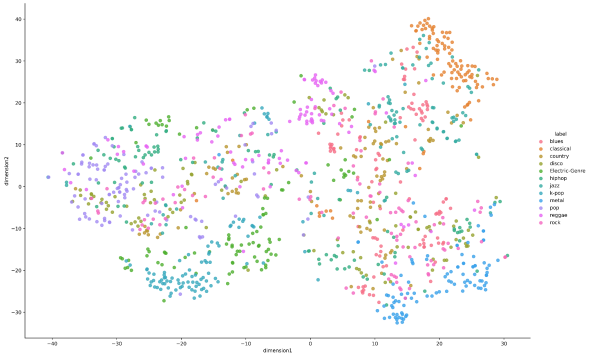

Figure 6: TSNE(n_components=2,n_iter=500) on original data

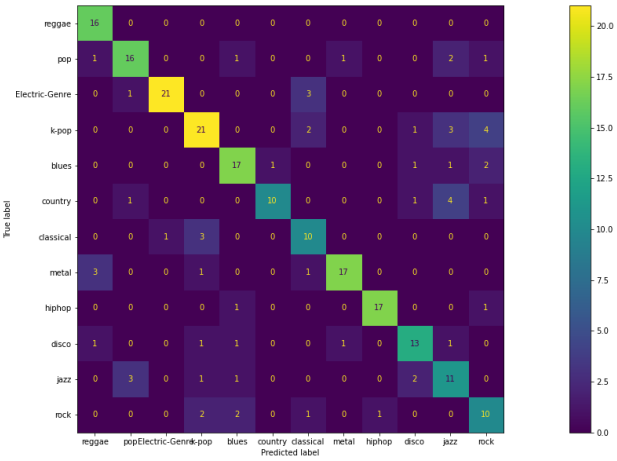

Figure 7: Confusion matrix of ANN for predicted and true labels

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_14 (Conv2D)           (None, 254, 254, 32)      896
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 127, 127, 32)      0
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 125, 125, 32)      9248
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 63, 63, 32)        0
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 62, 62, 32)        4128
_________________________________________________________________
max_pooling2d_16 (MaxPooling (None, 31, 31, 32)        0
_________________________________________________________________
flatten_6 (Flatten)          (None, 30752)             0
_________________________________________________________________
dense_12 (Dense)             (None, 64)                1968192
_________________________________________________________________
dropout_6 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_13 (Dense)             (None, 12)                780
=================================================================
Total params: 1,983,244
Trainable params: 1,983,244
Non-trainable params: 0
_________________________________________________________________
```

Figure 8: CNN model architecture used

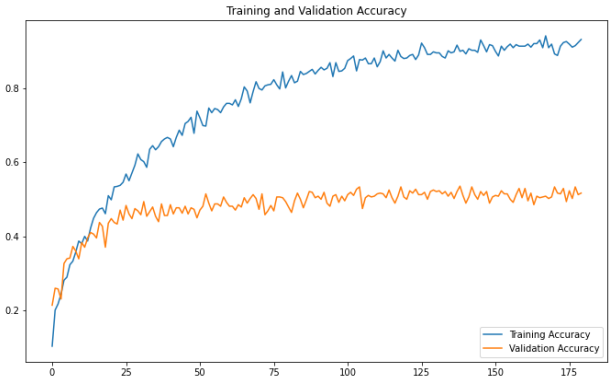

Figure 9: CNN - Varying validation and train accuracy with epochs (for analysis purpose) (is overfitted)

Figure 10: CNN - Varying validation and train loss with epochs (for analysis purpose) (is overfitted)

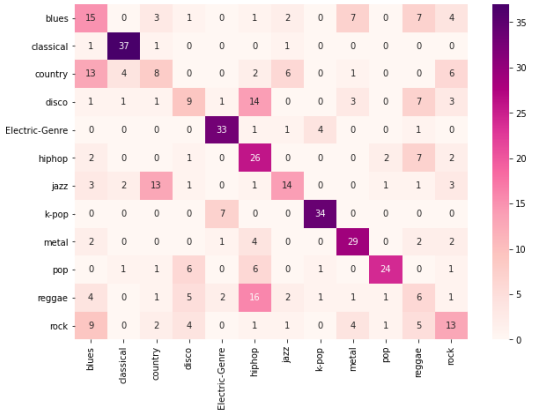

Figure 11: Confusion matrix of CNN for predicted and true labels

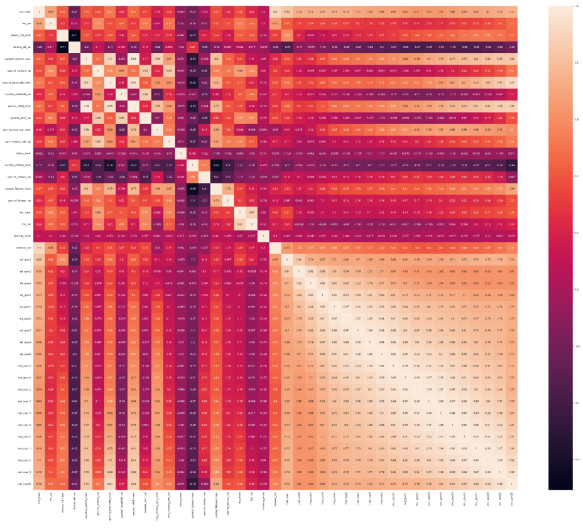

Figure 13: Heat map of initial data set 1200 rows, 43 column

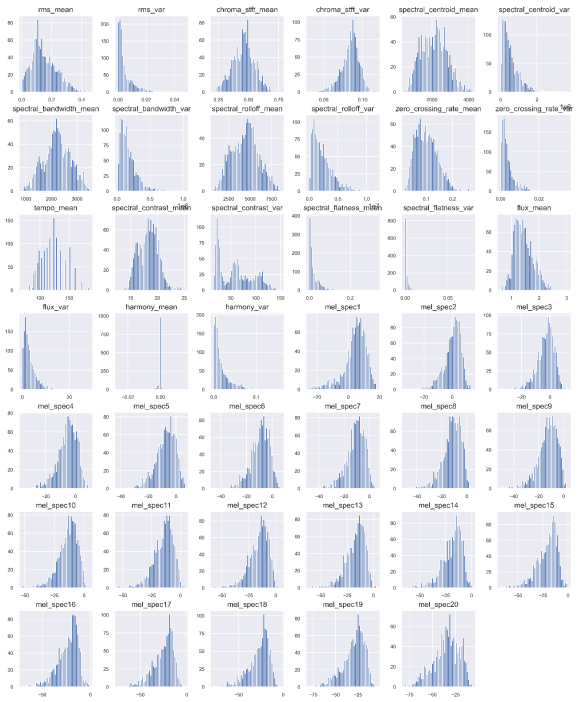

Figure 12: Histogram for all the features extracted using librosa

```
----------------------------------------------------------

Similar songs to blues.00002.wav

1 blues          -->  blues.00002.wav
                  Recommendation score = 0.9999998

2 rock           -->  rock.00009.wav
                  Recommendation score = 0.88283974

3 disco          -->  disco.00086.wav
                  Recommendation score = 0.87353134

4 country        -->  country.00072.wav
                  Recommendation score = 0.8699937

5 country        -->  country.00066.wav
                  Recommendation score = 0.8678256

6 blues          -->  blues.00080.wav
                  Recommendation score = 0.8677113

7 reggae         -->  reggae.00066.wav
                  Recommendation score = 0.86568165

8 blues          -->  blues.00010.wav
                  Recommendation score = 0.8656725

9 blues          -->  blues.00000.wav
                  Recommendation score = 0.86476517

10 rock          -->  rock.00064.wav
                  Recommendation score = 0.8637908

----------------------------------------------------------
```

Figure 14: Recommendation System for Ann using the classification model

```
If you like
 blues.00002.wav

 You might enjoy the following songs:

1 blues  -->     blues.00002.wav         with a distance of      0.0
2 rock   -->     rock.00011.wav          with a distance of      2.7803497824049
3 rock   -->     rock.00019.wav          with a distance of      3.001315490674597
4 hiphop  -->    hiphop.00070.wav        with a distance of      3.146432577232529
5 country  -->   country.00053.wav       with a distance of      3.180620099214508
6 k-pop  -->     02. Tamed-Dashed.wav    with a distance of      3.227529953712183
```

Figure 15: Recommendation System for KNN using K nearest neighbor on the best hyperparameter found on the classification model.

	precision	recall	f1-score	support
Electric-Genre	0.89	0.73	0.80	22
blues	0.73	0.89	0.80	18
classical	0.82	0.93	0.87	15
country	0.55	0.65	0.59	17
disco	0.45	0.53	0.49	19
hiphop	0.71	0.77	0.74	26
jazz	0.67	0.71	0.69	14
k-pop	0.86	0.86	0.86	21
metal	0.86	0.95	0.90	20
pop	0.86	0.82	0.84	22
reggae	0.75	0.65	0.70	23
rock	0.57	0.35	0.43	23
accuracy			0.73	240
macro avg	0.73	0.74	0.73	240
weighted avg	0.73	0.73	0.72	240

After applying Kfold, the accuracies of the model was observed to be - 0.698, 0.771, 0.729, 0.719, 0.729

Figure 16: Best hyperparameters for SVM - KFold accuracy, classification report.

```
Select song id(between 0 to 1199) you would like to listen:
134 ---> classical.00034.wav
801 ---> metal.00001.wav
1111 ---> rock.00011.wav
979 ---> pop.00079.wav
912 ---> pop.00012.wav
467 ---> press-pause-walz-main-version-01-38-17207.wav
586 ---> hiphop.00086.wav
684 ---> jazz.00085.wav
587 ---> hiphop.00087.wav
963 ---> pop.00063.wav
277 ---> country.00077.wav
1166 ---> rock.00066.wav
1057 ---> reggae.00057.wav
827 ---> metal.00027.wav
983 ---> pop.00083.wav
------------------------------------------------------------
277
0    classical.00053.wav
1         rock.00093.wav
2      country.00098.wav
3    classical.00054.wav
4      country.00066.wav
5    classical.00042.wav
6         rock.00048.wav
7         jazz.00066.wav
8      country.00072.wav
9         rock.00006.wav
Name: fileName, dtype: object
------------------------------------------------------------
Want more!!!!(y/n) : n
```

Figure 17 : Recommendation using Clustering Algorithms

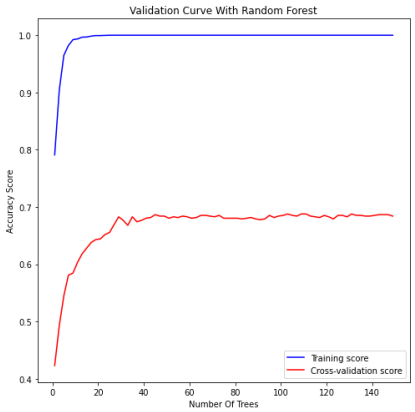

Figure 18: Validation Curve with basic Random Forest

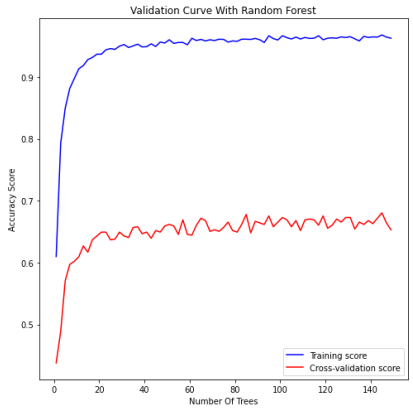

Figure 19: Validation curve with parameter tuned Random Forest

```
Model: "sequential_18"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_18 (Flatten)        (None, 41)                0

 dense_0 (Dense)             (None, 256)               10752

 normalize_0 (BatchNormaliza  (None, 256)              1024
 tion)

 dropout_0 (Dropout)         (None, 256)               0

 dense_1 (Dense)             (None, 128)               32896

 normalize_1 (BatchNormaliza  (None, 128)              512
 tion)

 dropout_1 (Dropout)         (None, 128)               0

 output (Dense)              (None, 12)                1548

=================================================================
Total params: 46,732
Trainable params: 45,964
Non-trainable params: 768
_________________________________________________________________
(1200, 128)
```

Figure 20: The model summary for ANN model