CSE 343: Machine Learning Interim 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. 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.

2. Related Work

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

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

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.

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] = (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. Analysis & Progress

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.

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 5) 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.

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).

In CNN, the model was chosen as it is the best model to extract audio and optical features from an audio file and their graphs. 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 (Figure16), the accuracy was 27% (Figure15). 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 10,11). 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 9,12,13,14). 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.

5. Results

In KNN, Logistic regression, CNN, and ANN results are shown in Table2. These results are obtained after applying preprocessing hyperparameter tuning.

In Clustering, we have used Silhouette Score[2] Davies Bouldin Score[3] to check clusters' separation (purity) instead of analyzing accuracy. (Table1). We used different techniques to increase the Silhouette score and decrease the Davies Bouldin score.

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.

In CNN, the details are shown from the confusion matrix in Figure9, the heatmap in Figure14, the training-test accuracy, and loss learning curves in Figures 10,12 and 11,13. Prediction is accurate, most for classical and worst for reggae genres. Figures 10 and 11 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). It also shows that the loss and accuracy are not interrelated by inverse relationship. Clustering and CNN accuracy were not much due to high sparsity in data.

In-state of art CNN [1], the dataset consists of 9991 audio files of 3s. It has two convolution layers with 256 kernels each (7x128) and relu activation, followed by a max pool layer. Dropout=0.5 is added. The gap is because of the difference in the number of layers, different input, and kernel dimensions (we are using 30s files). We have not currently used RNN/GRU in our model.

Considering the analysis of state of art KNN [12], the model was made, which gave good performance results (69 %). We tried a similar approach but not exactly the same and found similar results.

6. Future Work

For future work, we plan to study the application of various advanced methods to improve our RMSE scores for the supervised algorithms. We will try to reduce data sparsity (by including more data points or using new techniques) to solve the sparsity problem we are facing. For now, we have implemented the supervised and unsupervised learning algorithms and verified that the model is best fitted to the dataset. Now, we plan to mold our models to make a recommendation system and try to analyze the performance of our recommendation system. Then compare and contrast based on recommendations for each of these models. The ensemble approach will be implemented for EDA. In analysis, the Learning curve was used to analyze the model fit.

Future work division:

Siddharth:- Improve CNN model performance, Regression/Decision Tree, Recommendation system-CNN

Shanu:- Agglomerative, Hierarchical Clustering, Random Forest

Jasdeep:- SVM, Recommendation system - ANN, Repost making, loss minimization advance technique

APPENDIX (from this page onward)

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

Tables

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%

Table1: Hyperparameters, Silhouette Scores, and Davies Bouldin Scores for different clustering models

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

Table2: Accuracy, Precision, F1 and Precision scores for different models

Figures

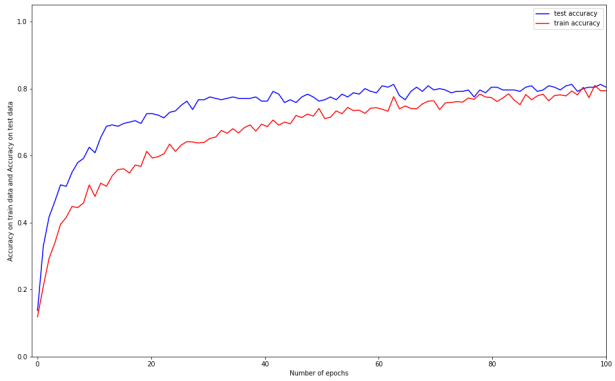

Figure 1: ANN - Varying test and train accuracy with epochs

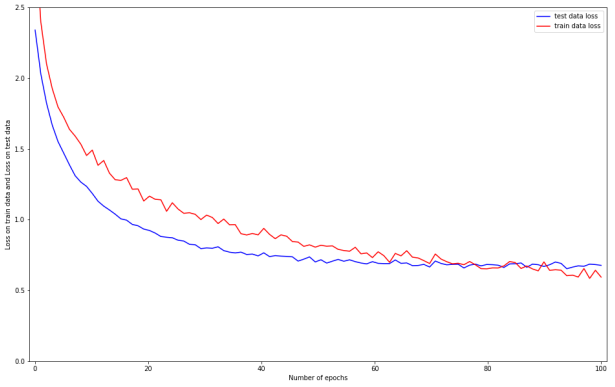

Figure 2: ANN - Varying test loss and train loss with epochs

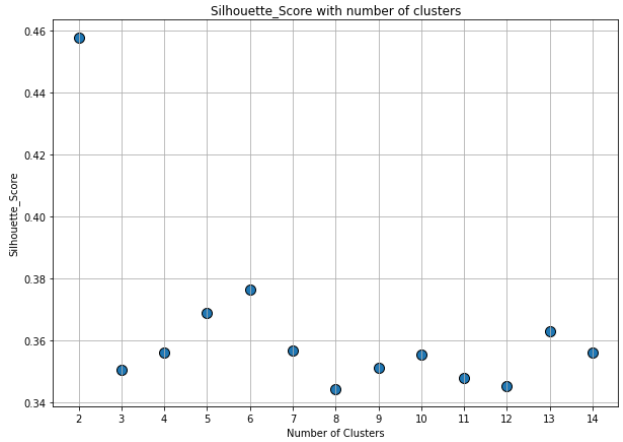

Figure 3: Silhouette_Score with number of clusters (K_Mean)

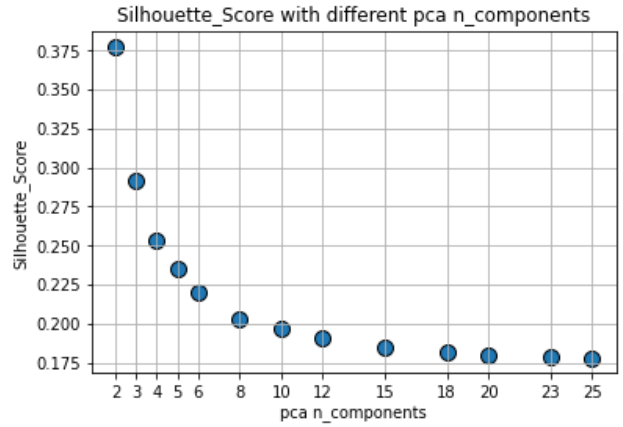

Figure 4: Silhouette_Score with different PCA components in K_Mean(n_clusters=9 n_init=9)

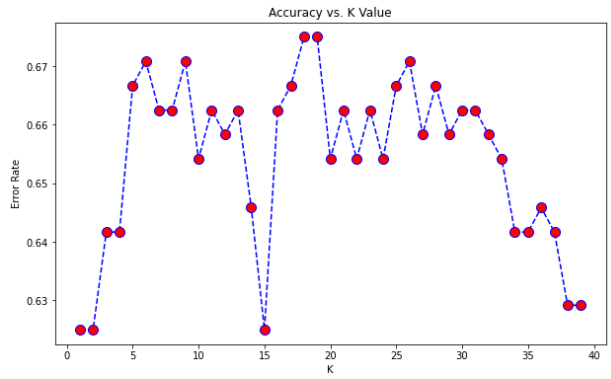

Figure 5: Accuracy varying with k number of neighbors

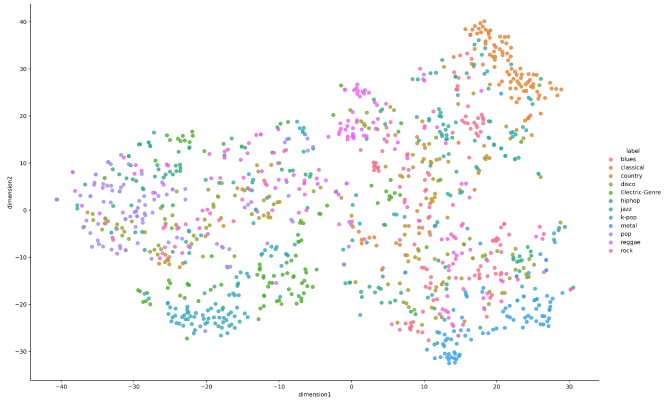

Figure 6: TSNE(n_components=2,n_iter=500) on original data

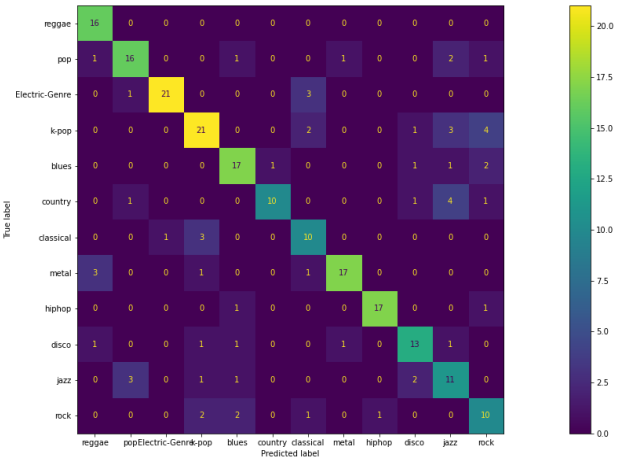

Figure 7: Confusion matrix of ANN for predicted and true labels

```
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

	precision	recall	f1-score	support
blues	0.30	0.38	0.33	40
classical	0.82	0.93	0.87	40
country	0.27	0.20	0.23	40
disco	0.33	0.23	0.27	40
Electric-Genre	0.75	0.82	0.79	40
hiphop	0.36	0.65	0.46	40
jazz	0.52	0.36	0.42	39
k-pop	0.85	0.83	0.84	41
metal	0.64	0.72	0.68	40
pop	0.83	0.60	0.70	40
reggae	0.17	0.15	0.16	40
rock	0.37	0.33	0.35	40
accuracy			0.52	480
macro avg	0.52	0.52	0.51	480
weighted avg	0.52	0.52	0.51	480

Figure 9: metrics obtained for each genre in CNN model

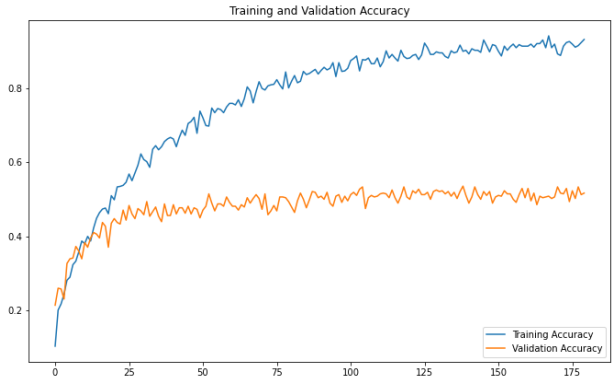

Figure 10: CNN - Varying validation and train accuracy with epochs (for analysis purpose) (is overfitted)

Figure 11: CNN - Varying validation and train loss with epochs (for analysis purpose) (is overfitted)

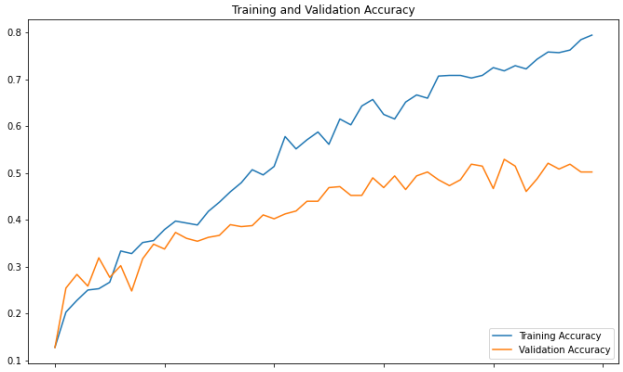

Figure 12: CNN - Varying validation and train accuracy with epochs (Final trained)

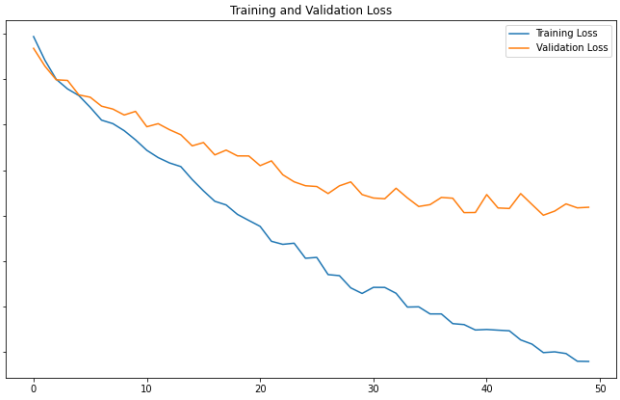

Figure 13: CNN - Varying validation and train loss with epochs (Final trained)

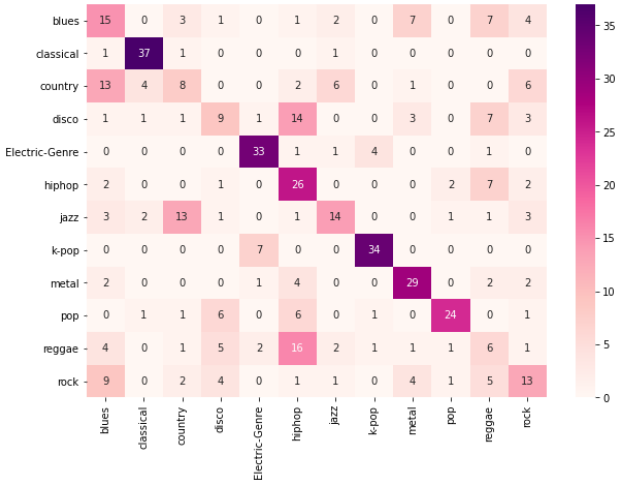

Figure 14: Confusion matrix of CNN for predicted and true labels

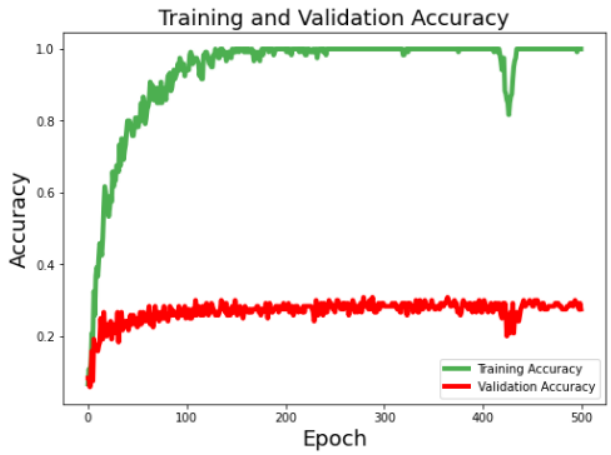

Figure 15: CNN model accuracy in initial stages

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 256, 256, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 128, 128, 32)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 128, 128, 32)      9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 64, 64, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 64, 64, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 32, 32, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 32, 32, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 65536)             0
_________________________________________________________________
dense (Dense)                (None, 128)               8388736
_________________________________________________________________
dense_1 (Dense)              (None, 12)                1548
=================================================================
Total params: 8,418,924
Trainable params: 8,418,924
Non-trainable params: 0
```

Figure16: CNN model in initial stages.

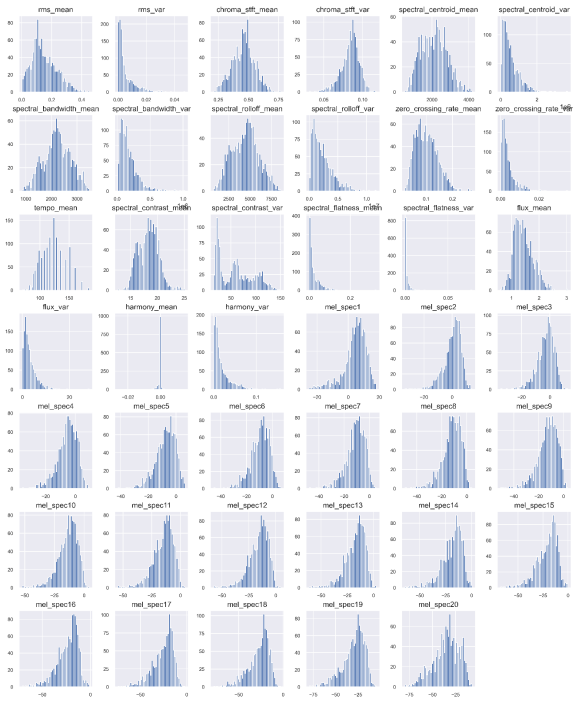

Figure17: Histogram for all the features extracted using librosa

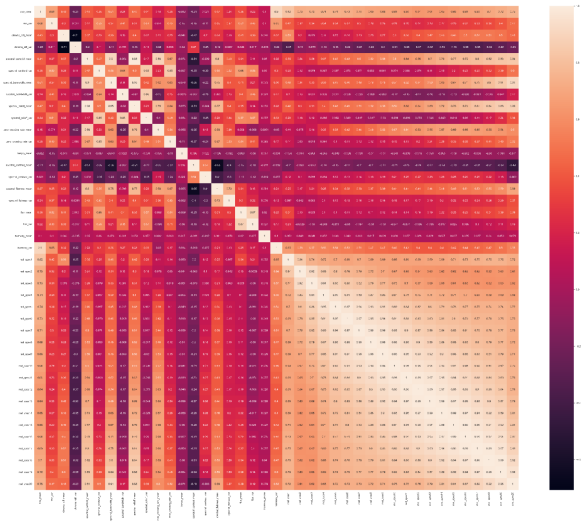

Figure18: Heat map of initial data set 1200 rows, 43 column