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

Final Report

Improvements on the Baseline and Intermediate models

KNN

While our Midsem submission relied on a custom written KNN model for classification, we have managed to further optimize the accuracy by using feature scaling on the in-built KNN provided by SKlearn using the feature_3_sec.csv file since this yielded the best results. The best hyperparameters were found using GrisSearchCV, tuning the **number of neighbours**, **weights** (distance-based or uniform) and the **method of distance calculation** (Manhattan/ Euclidean). While the best performance was achieved with the parameters**: number of neighbours = 1, weights = uniform, and metric = Euclidean**, the unity value of neighbours might indicate the model is overfitting the training data. Therefore, the value of K was chosen to be 2 or 3 as a trade-off between accuracy. As a result of these Improvements the test accuracy of KNN increased to **90.41%** with average cross validation accuracy of **89.7%.**

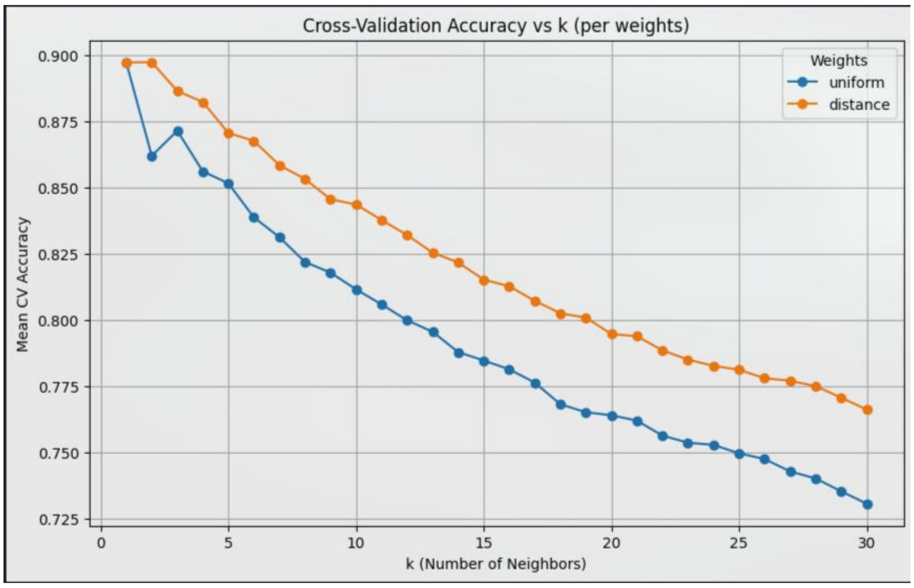
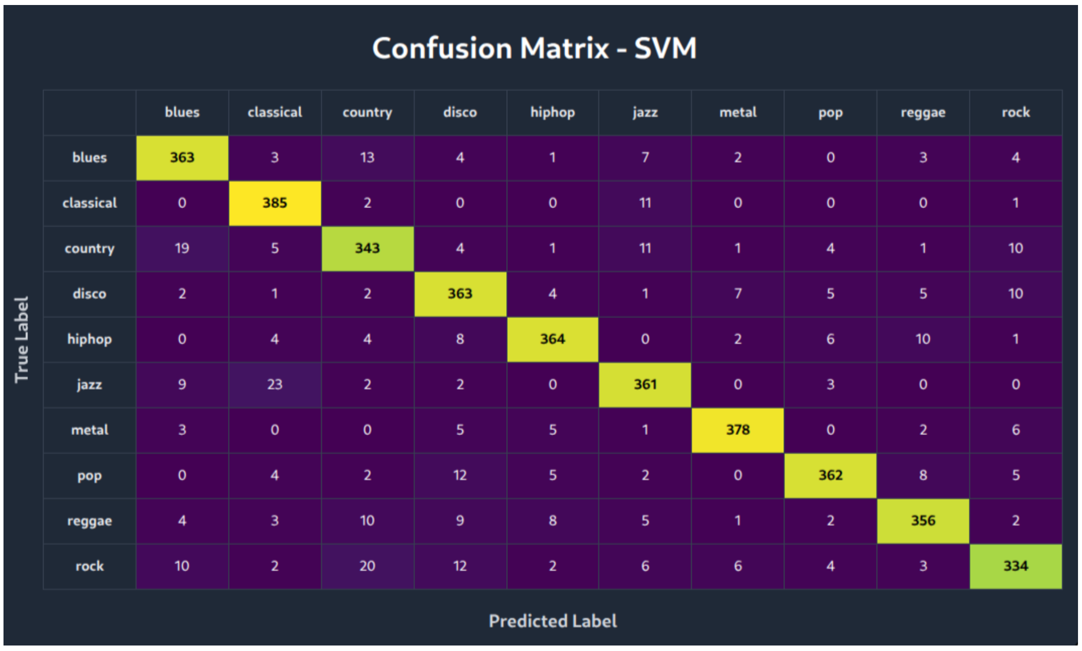


Fig:1 Results of Cross validation (5-fold) for KNN

SVM

As a part of our Stretch objectives, we also analysed the results through an SVM(Support Vector Machine) model. A pipeline was made using sklearn to include StandardScaler for feature normalization and SVC (Support Vector Classifier) from *scikit-learn* for model training. Hyperparameter tuning was performed using GridSearchCV with 5-fold cross-validation as we did earlier with KNN, with parameters including the **regularization coefficient** (*C*- Controls the trade-off between maximizing the margin and minimizing classification error), **kernel type** (rbf, polynomial, linear), **kernel degree** (only for polynomial kernels), **kernel coefficient** (*gamma, determines the influence of a single training example for RBF or polynomial kernels.*)**,** and **class weighting schemes**. The best performance was obtained using **the RBF kernel with C = 10, gamma = scale** (i.e., the value of gamma is set as per the variance of the data features)**, and balanced class weights**, achieving a cross-validation accuracy of approximately **89.2%** and a test set accuracy of **90.3%.** Since the RBF kernel yielded the best results, this indicated that the data was not linearly separable and most definitely required non-linear decision boundaries to reach optimal results.



	blues	classical	country	disco	hiphop	jazz	metal	pop	reggae	rock
blues	363	3	13	4	1	7	2	0	3	4
classical	0	385	2	0	0	11	0	0	0	1
country	19	5	343	4	1	11	1	4	1	10
disco	2	1	2	363	4	1	7	5	5	10
hiphop	0	4	4	8	364	0	2	6	10	1
jazz	9	23	2	2	0	361	0	3	0	0
metal	3	0	0	5	5	1	378	0	2	6
pop	0	4	2	12	5	2	0	362	8	5
reggae	4	3	10	9	8	5	1	2	356	2
rock	10	2	20	12	2	6	6	4	3	334

Fig :2 Confusion Matrix for SVM

Advanced Model

Convolution Neural Network (CNN):

The final model tested is a deep learning approach, ie, Convolution Neural Networks (CNNs). CNNs work very well with processing and learning patterns in grid-like data, such as images or audio spectrograms. They automatically learn spatial patterns and hierarchies of features through layers of convolutions, pooling and non-linear activations.

-Convolution is taking a filter that is sensitive to a particular local pattern and applying it to the grid-like data through a sliding window algorithm to produce the feature-map which highlights the presence of those local patterns.

-Pooling reduces dimensionality so that the filters do not have to slide over large grid spaces while still retaining important features and making the network more robust.

CNNs are extremely effective because they share weights across spatial locations in the grid space, thereby reducing the number of parameters compared to something like a densely connected Artificial Neural Network (ANN).

We experimented with multiple ways of implementing CNN models on our data ranging from custom made architectures to well-known architectures. However, it must be kept in mind that neural networks in general are extremely ‘data-hungry’ approaches and from that point of view, GTZAN is fairly small dataset with relatively very high number of features. This implies that the CNN has to learn too many parameters from a very small number of data points. This would most probably result in model-overfitting leading to low test accuracies.

We trained the following CNNs to compare the test accuracies:

1) **CNN (without regularisation):**
[Model accuracy=47.33%]

We tested this as a model to establish base performance benchmarks before iterating through tuning performance. The preprocessing for the model was fairly simple where we simply used Minimax Scaling on the MFCC coefficients for the **features_30_sec.csv** file. Further we split the data into 3 splits – the training set (70%), validation set (for parameter fine-tuning) and the test set (30%). Additionally, we had to add an extra dimension/channel to the scaled-dataset because of compatibility requirements of Keras for filters.

The model was fairly simple with 3 convolution blocks each of them using **ReLU** activations and padding the feature map each time. The output layer used **SoftMax** activation to get the likelihood of the data-point for each genre.

(It should be noted that we incorporated no normalisation or regularisation mechanisms to prevent model overfitting.)

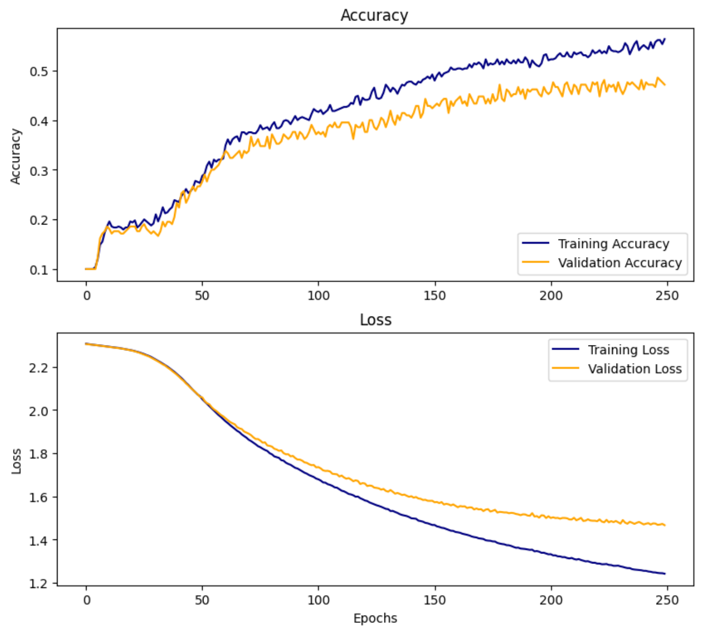


Fig :3 Change in Accuracy and Loss for cnn_1 over 250 epochs

2) CNN (with regularisation):
[Model Accuracy=53.66%]

This model had the exact same preprocessing as before. However, the architecture this time was somewhat different, i.e., we added dropout mechanisms for regularisation to prevent overfitting. **Dropout** is a mechanism wherein we randomly switch off certain neurons in every layer to prevent the model from relying on particular neurons a lot preventing the model from memorising the dataset. Additionally, we used **MaxPooling** on the feature map to introduce translation invariance (i.e.,small shifts/translations of the input wouldn't drastically change the output). Additionally, pooling also helps in preventing the model from learning noise, thereby having a regularising effect too.

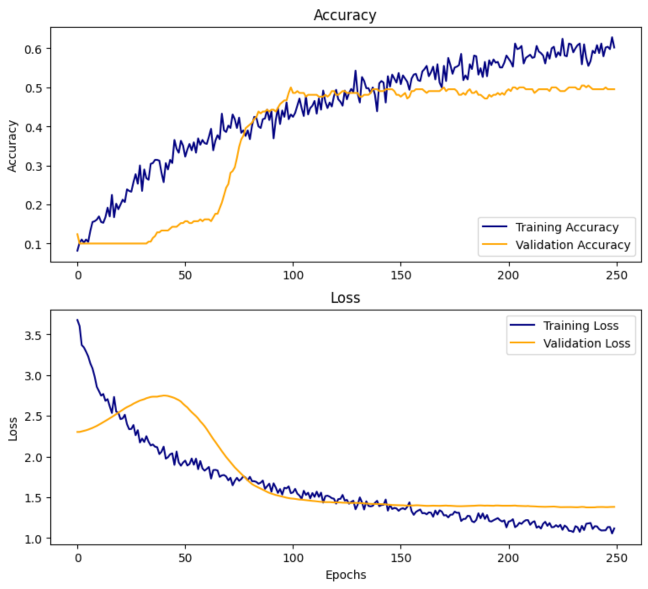


Fig :4 Change in Accuracy and Loss for cnn_2 over 250 epochs

<u>3)CNN (with regularisation and data augmentation):</u>

[Model Accuracy= 51.33%]

This model again had the same data-preprocessing. However, this time we added a technique called data augmentation. Data augmentation is a technique wherein we perturb the existing data to create additional data to train the model on in case of scarcity of training data. This also prevents overfitting as at times models can simply memorise the data if it's too small, hence augmentation helps the model generalise better by adding more diverse examples.

The way we augmented our data was, we simply flipped it since we are treating our grid-like csv data as an image thereby adding more diverse data for the model to train on. This could be analogous to feeding the audio data exactly in reverse.

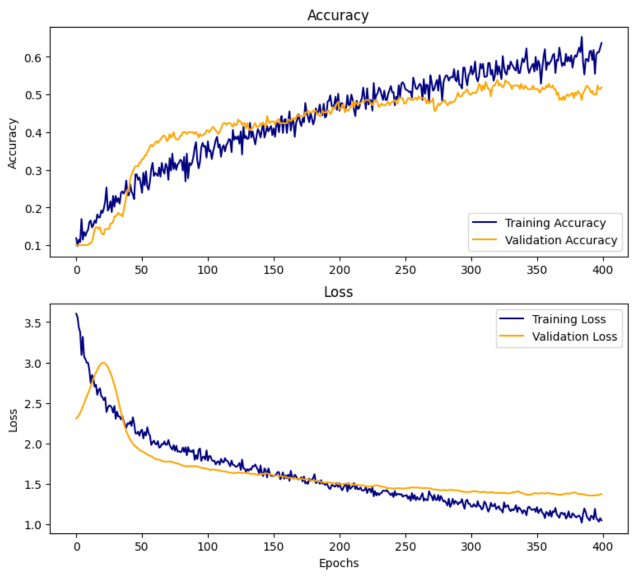


Fig :5 Change in Accuracy and Loss for cnn_3 over 400 epochs

Since, we were dissatisfied with the low model accuracies, we trained a few other CNN models:

a) A somewhat more complex model with 4 convolution blocks (accuracy=58%)

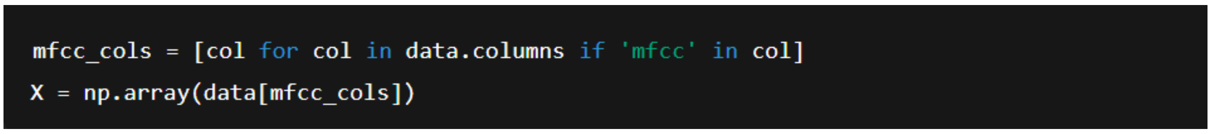
b) A model trained on Mel-Spectrogram data instead of MFCC coefficients, mel-spectrograms preserve more spectral details and temporal dynamics. (accuracy=58.9%)

c)A model with LeNet inspired architecture (accuracy=50.33%)

Innovations

1) There is a total of 58 MFCC coefficients in the **features_30_sec.csv** making the KNN algorithm extremely computationally intensive. Therefore, we added a Principal Component Analysis (PCA) layer on top of the existing mechanism to bring down the inference time for every test point **(from 632 milli-seconds to 9.39 milliseconds**). This was done by reducing the number of dimensions from 58 to 33 while still retaining 95 % variance. This however reduced the accuracy to 71% (perhaps we could run a hyperparameter finetuning on this to get the best hyperparameters, however here K was taken as 5).

2)We tried to make some changes to the preprocessing for the CNN (with regularisation) model that increased the accuracy to 67.6 %. We made the pre-processing clean and consistent such that the CNN receives the MFCC features in the correct order. The preprocessing earlier was something like this:



```
mfcc_cols = [col for col in data.columns if 'mfcc' in col]
X = np.array(data[mfcc_cols])
```

The problem with the code above is that pandas grabs all the columns with ‘mfcc’ as the heading but returns the features in alphabetical order (e.g., mfcc1_mean, mfcc10_mean, mfcc11_mean, … and so on) which jumbles the original order of the mfcc features so the patterns in the data get mixed up and the CNN learns the distorted patterns.

3) We further trained a 2-block CNN with HeUniform filter/kernel initializer paired with ReLU activation (with the same preprocessing as above). This is because ReLU suffers from vanishing gradients for negative values thereby reducing the variance in the output values for each neuron. However, HeUniform keeps this variance constant. (accuracy =67.3%)

Tasks Left

- We initially claimed that we could use the sparrow search algorithm (SSA) as a better alternative to the computationally intensive GridsearchCV for hyperparameter finetuning of the Random Forest model, however upon extensive training and research we concluded that the SSA algorithm is not only way more time consuming to implement and train but also yields hyperparameters worse than GridsearchCV (if not better).

- We could have implemented a boosting model also in addition to Random Forest (which is bagging based).

- We wished to build a proper recommender system/engine in addition to the genre classifier using unsupervised learning techniques.

- We also aimed at integrating a user interface to make our model somewhat high-level for a regular user.

Learning Experience

- We learned about different python libraries relevant for machine learning through this project, including niche ones like librosa for audio/speech features.
- We got a deep understanding of how majorly hyperparameters can affect the capabilities of even a simple model like (KNN).
- We learned that inference time also plays an important role in reducing the latency of production level systems like Spotify, Apple Music.
- We understood how to read, interpret and reproduce novel research work in the field of Machine Learning and Artificial Intelligence as a whole.
- We got a taste of team effort and collaborative effort.

Links:

- **Dataset**: https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification
- **Github Repo**: https://github.com/GeneraLIroh04/Course-Project-CS-F320-Foundations-of-Data-Science-BITS-Pilani-

___END___