Music Genre Classification: From k-NN to RNNs

Sébastien Pierre Marie Boo and Damian Herculano

TTT4275 Estimation, Detection, and Classification
Norwegian University of Science and Technology

April 29, 2024

Abstract

This report details the classification of music genres as part of a coursework project using the GTZAN dataset. To predict musical genres from audio features, we employed multiple machine learning techniques. For initial tasks, we implemented k-Nearest Neighbors (k-NN) classifiers to fulfill the assignment requirements. For the final task, we explored the potential of recurrent neural networks by implementing an ensemble of Long Short-Term Memory (LSTM) networks, which showed signs of increased accuracy compared to more conventional models. Project's code: https://github.com/seb2o/Music-genre-ml-classification.

Contents

1 Introduction **2**
- 1.1 Objectives . . . 2
- 1.2 The Datasets . . . 2
- 1.3 Data Splitting and Pre-Processing . . . 2

2 Theoretical Background **3**
- 2.1 Evaluation . . . 3
- 2.2 Machine learning models . . . 4
  - 2.2.1 k-NN . . . 4
  - 2.2.2 Random Forest . . . 4
  - 2.2.3 Neural Networks . . . 4

3 Task 1: k-Nearest Neighbors Classifier **5**
- 3.1 Methodology . . . 5
- 3.2 Results and Evaluation . . . 5

4 Task 2: Feature Distribution Analysis **6**
- 4.1 Methodology . . . 6
- 4.2 Results and Evaluation . . . 6

5 Task 3: Optimization through Feature Selection **8**
- 5.1 Methodology . . . 8
- 5.2 Results and Evaluation . . . 8

6 Task 4: Free-Choice Model Implementation **11**
- 6.1 Preliminary Exploration . . . 11
- 6.2 Model Specifications . . . 12
- 6.3 Results and Evaluation . . . 14

7 Conclusion **14**

1 Introduction

Music genre classification has practical applications in various digital media platforms, as it can enhance the user experience through personalized content. Our project tries to address this classification challenge by investigating different machine learning approaches.

1.1 Objectives

- To evaluate the effectiveness of k-NN models in music genre classification.
- To explore the impact of different audio features on classification accuracy.
- To explore and compare other machine learning techniques for robust classification.

1.2 The Datasets

The project utilizes the GTZAN dataset, known for benchmarking music genre classification. This dataset comprises 1000 audio tracks, each 30 seconds long, evenly distributed across ten music genres: pop, metal, disco, blues, reggae, classical, rock, hip-hop, country, and jazz.

Our work was actually based on three derivative datasets: *GenreClassData_5s.txt*, *GenreClassData_10s.txt*, and *GenreClassData_30s.txt*. They were created by segmenting each track into non-overlapping intervals of 5, 10, and 30 seconds respectively. These files contain 63 features extracted using the LIBROSA library, which include both statistical aggregates (mean and standard deviation) of fundamental audio signatures calculated from the track's spectrogram.

For task 1 to 3, we were required to only use the 30 seconds dataset. For task 4, we chose to only use the 5 seconds dataset, as justified in its corresponding section.

1.3 Data Splitting and Pre-Processing

The data was already partitioned into a training set of 800 tracks (80 from each genre) and a test set of 200 tracks (20 from each genre), ensuring uniform genre representation. Therefore, all we did when preparing the data before training was the following: First, we shuffled the dataset to make the training process less susceptible to the order of the data. Second, we split the data according to its designated 'Type'. Last but not least, we standardized the features by subtracting the mean and dividing by the standard deviation from the training set, ensuring unbiased and equal contribution from each feature.

For task 4, some more specific pre-processing was applied. This was key to our performance results and is further detailed in its corresponding section.

2 Theoretical Background

2.1 Evaluation

For the evaluation of a binary classification model, we can use a balance of precision and recall, namely, the Fβ-score.

- **Precision**: $\frac{TP}{TP+FP}$, that is the fraction of correctly identified samples among all the identified samples.
- **Recall**: $\frac{TP}{TP+FN}$, that is the fraction of correctly identified samples among all samples that should have been identified.

As seen in the class, the above notations TP, FP, and FN, refer to the nature of our observations, that is, "True Positive", "False Positive", and "False Negative" respectively. We can now use our definitions of precision and recall to introduce the **Fβ-Score**, which is given by

$$\text{F}\beta\text{-Score} = (1+\beta^2) \cdot \frac{precision \cdot recall}{\beta^2 \cdot precision + recall}.$$

where β weights the importance of *recall* with respect to *precision*. In the case when $\beta = 1$, we obtain what we refer to as the **F1-Score**:

$$\text{F1-Score} = \frac{2 \times TP}{2 \times TP + FP + FN}$$

However, when dealing with m-ary classification, we cannot directly compute those metrics for the whole model because we need to average them in some way to represent the performance of the model for each class. The approach we have chosen to pursue is to consider the "1 vs all" approach with the *macro* average. That is, for each class, we consider it as *true* label, and all the other are considered $false$. Then, we compute metrics as if it were binary classification and take the mean. This can be expressed as:

$$\text{Macro Average} = \frac{1}{N} \sum_{i=1}^{N} \text{Metric}_i$$

where N is the number of classes, and Metric_i is the computed metric for the i-th class. This way works well for balanced datasets because it is simpler than micro average, at the cost of erasing class distribution in the error calculation.

Thus, to evaluate the overall model performance, we will frequently use the **Macro F1-Score**, defined as:

$$\text{Macro F1-Score} = \frac{1}{N} \sum_{i=1}^{N} \text{F1-Score}_i$$

This metric will often be referred to when discussing the "Accuracy" of a model as it provides a somewhat better picture of the model's performance, since the "Accuracy" is simply defined as:

$$\text{Accuracy} = \frac{\text{number of correct predictions}}{\text{total number of predictions}}$$

Also, an easy way to gauge the performance is to look at the confusion matrix, which we will display multiple times in this report. The cell M_{ij} is the number of samples of

label i classified as label j. Thus the ideal CM is diagonal. Since we do not accord more importance to Type I errors than to Type II errors, there is no other consideration to take into account and the CM yields a very effective representation of the model's performance.

2.2 Machine learning models

All models used in this project are examples of supervised learning. State-of-the-art (SOTA) for this task typically involves unsupervised learning followed by fine-tuning; however, this approach requires significantly more data than was available to us. Down below, you will find a very broad introduction to some of the machine learning models we refer to in this report.

2.2.1 k-NN

The k-Nearest Neighbors (k-NN) algorithm is a simple non-parametric method used in classification tasks. A new point is classified by a majority vote of its neighbors, with the point being assigned to the class most common among its k nearest neighbors measured by a distance function. The distance is typically calculated using Euclidean distance, although other metrics can be used depending on the problem context. The k-NN algorithm is particularly effective in scenarios where the decision boundary is irregular, as it does not assume any probability distribution on the input data.

2.2.2 Random Forest

In Task 3, we employed a Random Forest classifier to optimize our feature set and evaluate feature importance. Random Forests work by creating multiple decision trees and determining the class output based on the most common result from these trees.

In our case, the crucial aspect of using Random Forests is their method of assessing feature importance. The model estimates how significantly each feature reduces impurity across all decision trees, where impurity means Shannon entropy. By averaging these reductions, we were able to identify which features have potentially the most impact on our classifiers' accuracy.

2.2.3 Neural Networks

In Task 4, we explored the use of neural networks, which are great tools for modeling complex patterns in data. They consist of layers of interconnected nodes, or neurons, where each connection represents a weighted dependency between nodes.

For our project, we specifically used a Recurrent Neural Network, based on Long Short-Term Memory (LSTM) layers. Given a sequence $\boldsymbol{\{x_t\}_{t=0}^{t=T}}$, the RNN outputs a result for each timestep, depending on the current time and the previous output : $RNN_t = f_\theta(x_t, RNN_{t-1})$, where θ is a set of parameters learnt via gradient descent.

3 Task 1: k-Nearest Neighbors Classifier

3.1 Methodology

The first task was relatively straight-forward. We implemented a k-NN classifier with the 'sklearn' library using the four features specified in the project brief: spectral rolloff mean, MFCC 1 mean, spectral centroid mean, and tempo. The choice of k was set to 5, as required by the assignment. In addition, we employed 'distance' weighting for our classifier to prioritize nearer neighbours.

3.2 Results and Evaluation

We quantified the k-NN classifier's effectiveness using a confusion matrix and F1-Scores for each genre. Overall accuracy and macro F1-Score were also computed.

Shown below is the confusion matrix, as well as the F1-scores by genre.

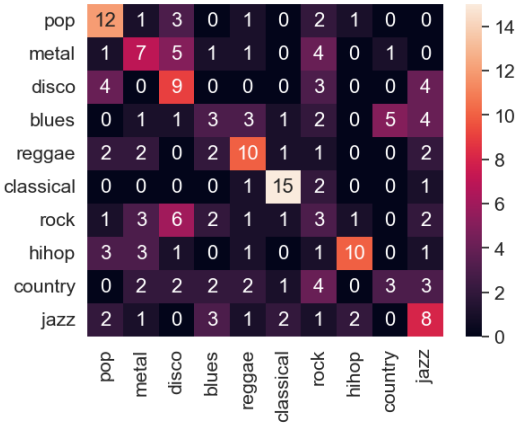

Figure 1: Confusion Matrix

Genre	TP	TN	FP	FN	F1-Score
Pop	12	165	13	8	0.533
Metal	7	165	13	13	0.350
Disco	9	160	18	11	0.383
Blues	3	168	10	17	0.182
Reggae	10	167	11	10	0.488
Classical	15	173	6	4	0.750
Rock	3	158	20	17	0.140
Hip-Hop	10	174	4	10	0.588
Country	3	173	6	16	0.214
Jazz	8	161	17	12	0.356

Figure 2: F1-Scores by Genre

The above analysis provides us with insights into the performance of our classifier. We note that the model performed relatively well for Classical music, but performed poorly otherwise, especially when it comes to Rock, Blues and Country music.

Overall, the model's accuracy was 40.4% and its macro F1-score was 0.398. These two metrics need to be contextualised with the high discrepancy in accuracy among genres. Also, we need to deepen our understanding of the datasets features distribution, which is the objective of the next task.

4 Task 2: Feature Distribution Analysis

4.1 Methodology

This task involved a statistical analysis of feature distributions across four genres: pop, disco, metal, and classical. Specifically, we analyzed the distributions for four key features: spectral rolloff mean, MFCC 1 mean, spectral centroid mean, and tempo.

4.2 Results and Evaluation

As seen below, we plotted the distributions of the four features across the selected genres using histograms to visualize their density.

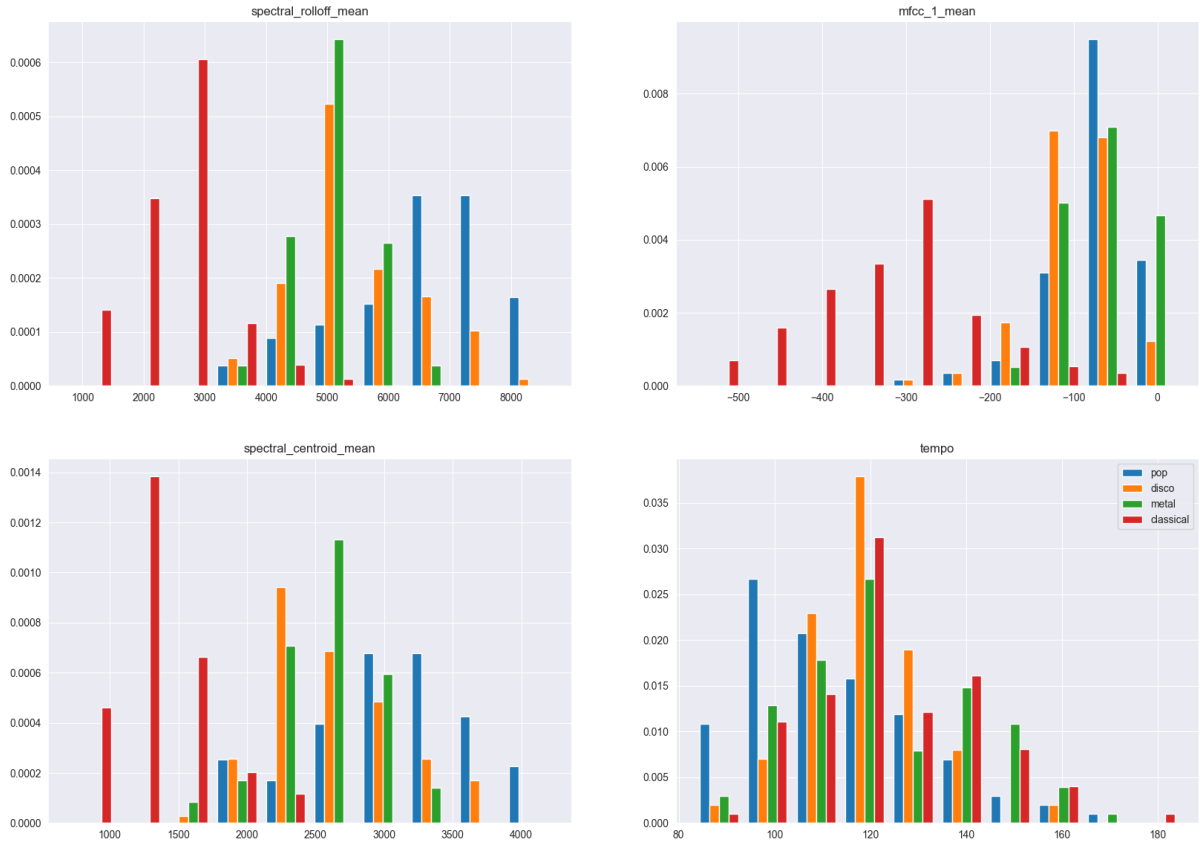

Figure 3: Histograms showing the distribution of spectral rolloff mean, MFCC 1 mean, spectral centroid mean, and tempo across four music genres.

From the histograms, we can make the following observations:

- **Spectral Rolloff Mean**: This feature shows considerable variation between classical, represented by the lower values, and pop, spread over the larger values with a distribution slightly skewed to the right. Disco and metal are in-between, showing a similar distribution.
- **MFCC 1 Mean**: Once again, classical music stands out here. Its values are significantly lower compared to the other three genres, which show a relatively similar distribution with lower variance (narrower distribution).

- **Spectral Centroid Mean**: Similarly, classical music clearly differentiates itself. Its values are lower and it has a narrower distribution. The other three genres are hardly differentiable, despite pop showing a wider distribution with a mean slightly higher in value.

- **Tempo**: This time, pop music seems to be standing out the most. Its distribution is notably skewed to the left, towards lower values. The other three genre show a similar distribution.

These observations are completely inline with our results obtained in task 1. Classical music, our highest performing genre, is standing out the most over these features. Our classifier was apparently able to capture those distributions relatively effectively. Regarding Pop music, our results from task 1 are also consistent with our observations. This genre had the 3rd highest F1-score, close behind Hip-Hop, as it shows some distinguishable patterns, especially in tempo and spectral rolloff mean. Whereas for Metal and Disco, we should not be surprised of their poor results in task 1. They are hardly distinguishable on these four features. This is the reason why we were excited to tackle task 3, where we will eventually be able to increase our overall accuracy by selecting a different feature combination.

5 Task 3: Optimization through Feature Selection

5.1 Methodology

The task assigned was to strategically choose a fourth feature to add alongside 3 features within a group of 4 pre-defined features. Those 4 features are the ones examined in the previous task: *spectral_rolloff_mean*, *spectral_centroid_mean*, *mfcc_1_mean*, and *tempo*. To begin our analysis, we examined the correlation between all features. However, the insights from the correlation matrix were insufficient for a conclusive feature selection. To go further, we decided to implement a Random Forest classifier based on the training dataset across all features. We did so to be able to evaluate each feature's individual impact using a feature importance score. The aim was to identify which features contribute the most to the classifier's performance.

Despite identifying highly influential features, we recognized that the best individual features might not necessarily enhance the model's accuracy, especially given that there are multiple ways for choosing the initial 3 features in the first place. Therefore, we ultimately decided to evaluate all possible combinations composed of three base features augmented by an additional feature. Since there are 4 ways of choosing a three-element subset in a set of size 4 and there are 59 potential additional features, we get a total of $\binom{4}{3} \times 59 = 4 \times 59 = 236$ combinations.

5.2 Results and Evaluation

Below, you can find the result of our feature importance analysis. That is, the resulting contribution of each feature to a random forest classifier trained on all features.

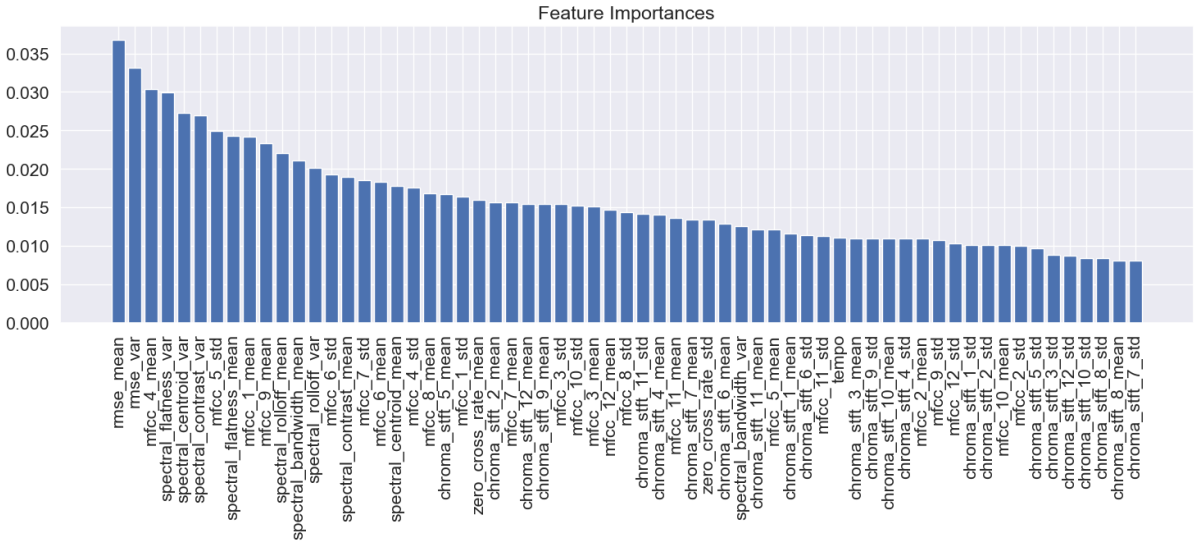

Figure 4: Feature importances from the Random Forest classifier.

This plot allows us to shortlist a number of highly influential features. Namely, *rmse_mean*, *rmse_var*, *mfcc_4_mean*, and *spectral_flatness_var*. However, as mentioned earlier, this does not provide us with any information on how well these features may complement our initial set of features. Therefore, we have trained and tested a k-NN classifier, similarly to task 1, for all possible combinations and plotted their corresponding macro F1-score as a heatmap.

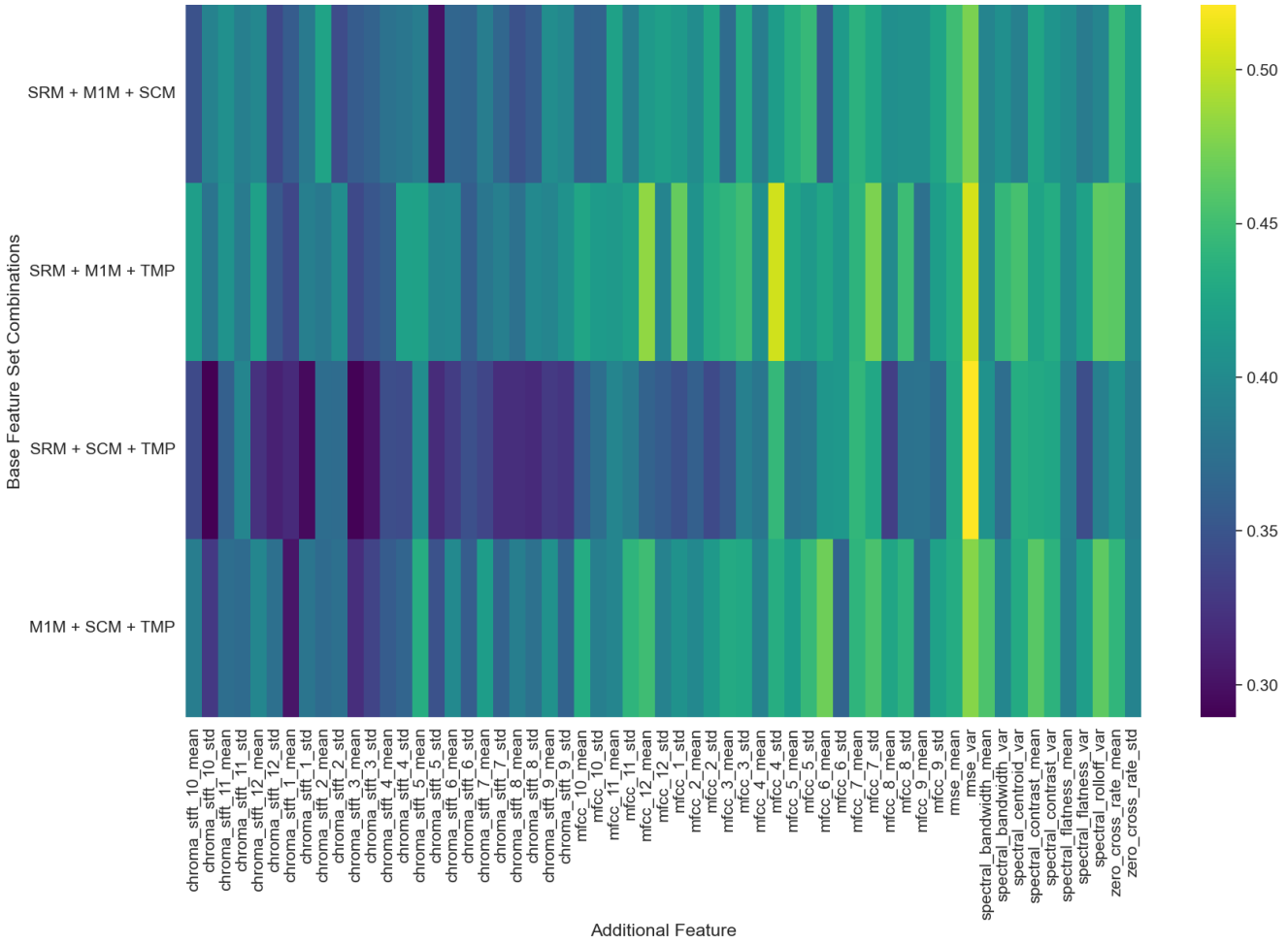

Figure 5: Heatmap of macro F1-scores for all feature combinations. Note: (TMP) *tempo*, (SRM) *spectral_rolloff_mean*, (SCM) *spectral_centroid_mean*, and (M1M) *mfcc_1_mean*

The best performing combination included the following three base features *spectral_rolloff_mean*, *spectral_centroid_mean*, *tempo*, and the additional feature *rmse_var*, which was interestingly part of our importance shortlist. We also note that the *chroma* features on the left hand side of the heatmap tend to be less relevant. This is once again consistent with our feature importance experiment, where those features were mainly represented in the right hand part of the importance plot. The actual performance of our winning combination is displayed below:

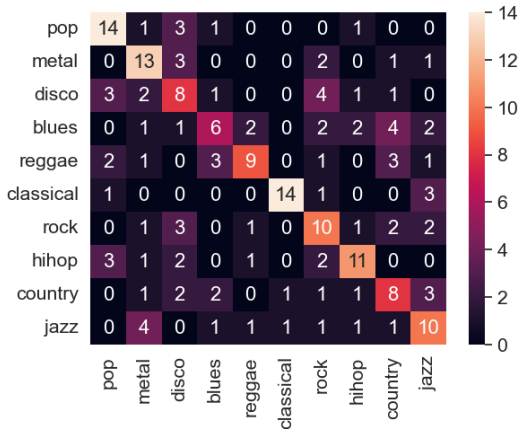

Figure 6: Confusion Matrix Task 3

Genre	TP	TN	FP	FN	F1-Score
Pop	14	169	9	6	0.651
Metal	13	166	12	7	0.578
Disco	8	164	14	12	0.381
Blues	6	170	8	14	0.353
Reggae	9	173	5	11	0.529
Classical	14	177	2	5	0.800
Rock	10	164	14	10	0.455
Hip-Hop	11	171	7	9	0.579
Country	8	167	12	11	0.410
Jazz	10	166	12	10	0.476

Figure 7: F1-Scores by Genre Task 3

The newly constructed confusion matrix and F1-Score table show a significant increase in performance compared to our evaluation in task 1. In fact, we've improved the individual F1-Score of nearly all genres. It is especially notable for our previously worst performing genres, which had F1-Scores of 0.140 (Rock), 0.182 (Blues), and 0.214 (Country).

Overall, this feature set achieved an accuracy of 52.02% and a macro F1-score of 0.521. As a reminder, our evaluation of task 1 resulted in an accuracy of 40.4% and a macro F1-score of 0.398. This improvement over task 1 reminded us of the importance of feature selection. But how about the importance of model selection ? Parts of our final task, where we explore deep learning capabilities, attempt to answer this very question.

6 Task 4: Free-Choice Model Implementation

6.1 Preliminary Exploration

For the final task of this project, we were free to choose the model type, the number of features and the datasets to use. Given the extensive set of features, the complex inter-dependencies among them, and no fewer than 10 classes, we quickly realized that deep learning models would likely outperform simpler classification models such as K-NNs. Recognizing this, we were initially uncertain about which architecture would be most effective. Also, since deep lerning requires a lot of data, we initially used a concatenation of the 3 datasets and considered all 63 features. Consequently, we began our exploration by testing a variety of models with very minimal data pre-processing, across four categories: Fully Connected Neural Networks (FCNNs), Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Hybrid Models. It turned out that our hybrid models performed quite poorly: they seemed over-engineered and largely overfitted the training data. The other models yielded better results, with accuracies ranging between 67% and 72%. For the specifications of all our models, please refer to the corresponding *task4_exploration.ipynb* notebook.

Thinking a bit more about the structure of the datasets, we realised that concatenating the datasets was not the best idea, as the structure of each sample would vary depending on whether its features were measured during 5, 10 or 30 seconds, thus hindering learning. Using df30s seemed to yield the best results but after some more thinking, we decided that measuring the features during 30 seconds and evaluate their mean surely leads to information loss. However, using each of df5s samples separately yielded bad results. Then, we realized that, since the model should be classifying all samples of the same track with the same label, we should select, for each track, the majority genre predicted among its samples. For example, in the 5 seconds dataset, each track was divided into 6 samples. If the model predicts two of the samples to be rock, and 4 to be metal, then we set all samples to be classified to metal. We evaluated the impact of this method in *task4_is_vote_good.ipynb*. It yielded an improvement in accuracy of 5%, effectively being better than using plain df30s.

Still, there was some consequent information loss in the process since the samples were treated independently of each other at training. This can be seen as losing the temporality of the music. Indeed, it would make sense that, for instance, introductions of rock and metal musics are very similar, but the middle of the music differs a lot. Thus, we turned to recurrent neural networks to classify each track. We describe the selected model peculiarites in the next section.

6.2 Model Specifications

Below are listed some of our model specifications and observations.

- We worked using only the 5 seconds (df5s) dataset.
- We implemented 1 layer of LSTM right before a Dense layer, with some dropout before the output layer.
- The LSTM offered the best results, a SimpleRNN was faster but performed slightly worse, and the BI-LSTM architecture was both worse and took longer. Any combination of more than 1 recurrent layer caused overfitting, even with very few units.
- 128 units in the LSTM layer was chosen as the best performing number of units compared to 512, 256 and 64.
- The Dense layer was found to perform better when having the same number of unit as the LSTM layer, and the dropout was found by trial and error.
- The activation functions we selected were *ReLu* for the Dense Layer and *softmax* for the output layer. The optimizer chosen was *Adam*. These choices were made by selecting the best performing among a small subset of other possibilites; namely, *tanh* and *sigmoid* for the activation functions, and *RMSProp* for the optimizer.
- We had to reshape the data into a list of sequence: initially, we had 5940 samples, each consisting of 63 features. We then group the samples by their track, and keep the 40 most important features, selected by $MRMR$. The final shape is (990, 6, 40) : 990 tracks, each constituted of 6 samples, one sample being a vector of 40 features. During the process of reshaping, we also shuffle the order of the tracks.

The MRMR technique, similar to an approach used by Uber engineers, is a systematic feature selection process. The idea is to identify features with *Maximum Relevance and Minimum Redundancy*, using correlation with target and pairwise correlations. This allows for a faster training and fewer risks of overfitting.

Last but not least, the ensemble approach was chosen to further improve accuracy, effectively combining 10 models. This was done to mitigate any model-specific biases or variances, where each model in the ensemble would theoretically correct the errors of its peers. Therefore, the ensemble's final prediction for each music track was determined by a majority vote among the 10 individual models' predictions. We did so to ensure that our outputs were both robust and reliable. It is important to distinguish sample voting, where a track genre is voted by each of its samples genre, from model voting, where each track genre is voted by several different models.

The following figure illustrates the final architecture of our LSTM model ensemble, where each LSTM processes the input sequences independently, and their predictions are aggregated through majority voting.

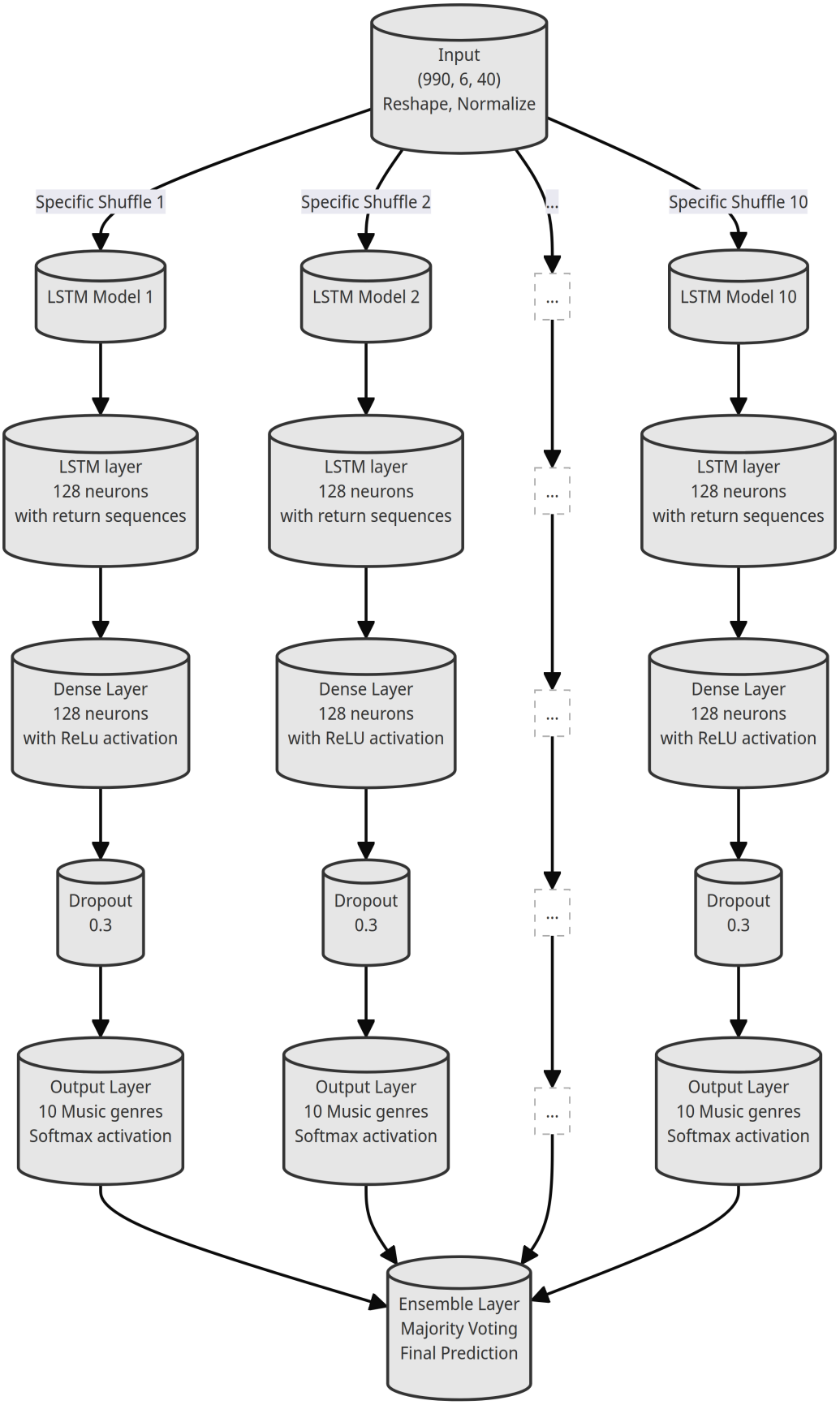

Figure 8: Architecture of the LSTM ensemble model.

6.3 Results and Evaluation

In a similar manner to what we've previously done, the LSTM ensemble's performance was assessed through its resulting confusion matrix, the F1-Scores for each genre, and its overall accuracy and macro F1-Score.

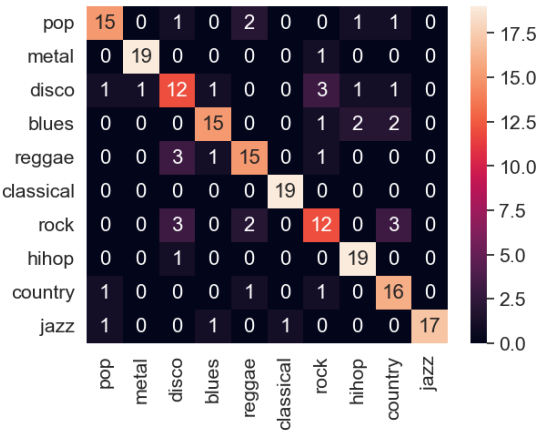

Figure 9: Confusion Matrix for Task 4

Genre	TP	TN	FP	FN	F1-Score
Pop	15	175	3	5	0.789
Metal	19	177	1	1	0.950
Disco	12	170	8	8	0.600
Blues	15	175	3	5	0.789
Reggae	15	173	5	5	0.750
Classical	19	178	1	0	0.974
Rock	12	171	7	8	0.615
Hip-Hop	19	174	4	1	0.884
Country	16	172	7	3	0.762
Jazz	17	178	0	3	0.919

Figure 10: F1-Scores by Genre for Task 4

Our results indicated a significant success with the ensemble approach. In fact, our 10 individual LSTM models obtained a mean accuracy of 76.9%. However, our resulting ensemble model achieved, as a whole, an accuracy of 80.3% and a macro F1-Score of 0.803. The above confusion matrix and table also show a relatively good consistency among genres, with Rock and Disco being the worst performers with an F1-Score of 0.600 and 0.615 respectively. This is a big step forward compared to our classifiers from the previous tasks. We are especially pleased to have achieved near perfect F1-Scores for Classical, Metal, and Jazz at 0.974, 0.950 and 0.919 respectively.

7 Conclusion

In conclusion, we investigated different machine learning models for classifying music genres using the GTZAN dataset. We began with k-Nearest Neighbors classifiers, achieving a baseline accuracy of 40.4% and a macro F1-score of 0.398. Our exploration of feature distributions and selective feature optimization in subsequent tasks improved our model's performance to a macro F1-score of 0.521 and accuracy of 52.02%.

Finally, we implemented an ensemble of 10 Long Short-Term Memory (LSTM) networks, which led to a further enhancement in performance. This ensemble approach demonstrated a clear advantage on our dataset, providing more robust results compared to the more traditional deep learning models we've evaluated.

Overall, this project was very exciting. Given that we had worked multiple times on image classification problems in previous classes, we were really keen on working with this music dataset, and it did not disappoint. The assignment also facilitated our initial foray into more sophisticated models, particularly through our exploration of recurrent neural networks. As we progress in our studies, the experience gained from this complex classification problem is greatly valuable, and we look forward to building on this experience for future challenges.