Comparative Analysis of Machine Learning Approaches for Music Genre Classification: Audio Samples, Lyrics, and Audio Features

Anjali Pudiyadatha [G01389954]
Madhan Balaji Rao [G01374078]
Balassubramanian Srinivasan [G01396498]

1 Abstract

Music genre classification is an important task in music information retrieval. The goal is to classify music tracks into one or more of the many genres. Classification of tracks into genres can help in organizing music collections, creating playlists, and providing personalized music recommendations. In this project we aim to develop and compare machine learning models to classify tracks into genres through multiple distinct approaches, using a 30s sample of the track, using the lyrics and using audio features retrieved through Spotify's API. We plan to explore various machine learning algorithms and evaluate their performance using various metrics such as accuracy and F1 measure. Additionally, we will investigate the impact of different audio feature extraction techniques on the performance of the model.

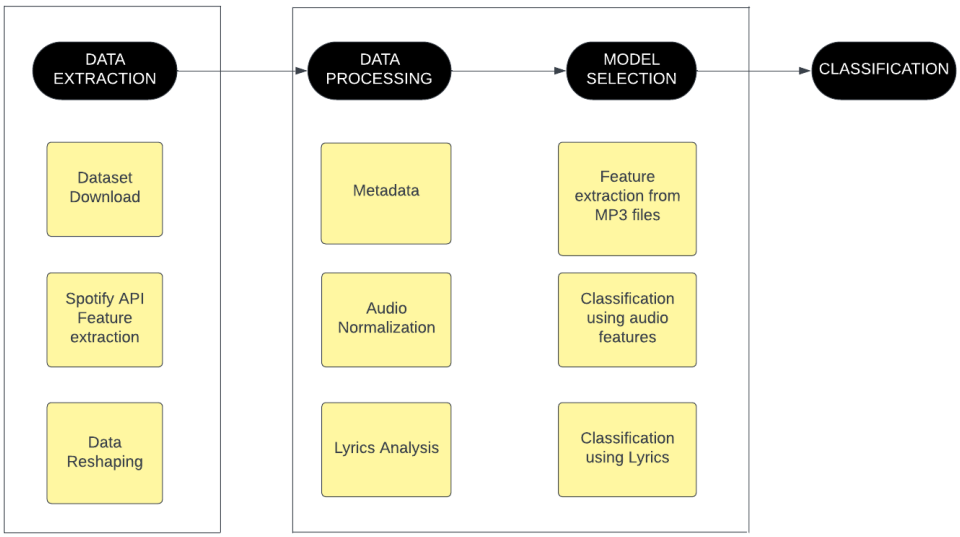

Figure 1: Process Flow

2 Introduction

Music genre classification is a critical aspect of music information retrieval, as it aims to categorize music tracks into one or more distinct genres. The ability to accurately classify tracks by genre holds significant implications for organizing music collections, generating playlists, and delivering personalized music recommendations. In light of the increasing importance of this task, the primary motivation behind this project is to develop robust machine learning models that can effectively classify music tracks into their respective genres using multiple approaches. The problem at hand consists of determining the most effective method for music genre classification, given the varying nature of the data sources and potential limitations associated with each approach. To address this problem, we plan to explore and compare three distinct methods: analyzing 30-second MP3 samples of tracks examining the lyrical content of tracks, and utilizing audio features of tracks retrieved through Spotify's API. By employing a diverse set of machine learning techniques tailored to each approach, such as neural networks for audio samples, natural language processing models for lyrics, and support vector machines for audio features, we aim to identify the most effective and accurate method for music genre classification. Through this comprehensive investigation, we strive to create a powerful classification model that caters to a wide range of music genres and user preferences, ultimately enhancing the overall music listening experience.

3 Problem Statement

The primary objective of this project is to develop effective music genre classification models for a data set of approximately 5,000 tracks, utilizing three distinct approaches. The data set consists of tracks along with their audio features and 30-second MP3 samples, both obtained through Spotify's API, and their corresponding lyrics, which are retrieved via LastFM. The first approach will require the implementation of a machine-learning model that can process and analyze the 30-second MP3 samples to classify the tracks into genres. This model may involve the use of deep learning techniques, such as convolutional neural networks (CNNs), to capture the intricate patterns present in the audio signals. However, challenges may arise in handling the variations in audio quality and the presence of noise in the samples. In the second approach, a NLP-based model, such as a transformer or recurrent neural network (RNN), will be employed to analyze the lyrical content of the tracks for genre classification. This method seeks to identify patterns and themes that are characteristic of specific genres. The potential challenges include the overlapping of themes across genres and the presence of ambiguous or nonspecific lyrics. Lastly in the third approach in order to accomplish our goal, we initially process the data by scaling certain features to enhance the effectiveness of the model training process. We proceed to utilize a wide range of classifiers, with each classifier being optimized through a combination of grid search and cross-validation. Lastly, we assessed the performance of each classifier on the test data, examining various metrics such as accuracy, F1 score, and confusion matrices. The project aims to compare the effectiveness and accuracy of these three approaches in classifying music genres while addressing the potential challenges associated with each method. By employing diverse data sources and machine learning techniques, we strive to create a comprehensive and versatile classification model that caters to a wide range of music genres and user preferences.

4 Related Work

Music genre classification has been an active area of research in music information retrieval, with various studies exploring different techniques to enhance the accuracy and efficiency of classification models. A prominent study conducted by Tzanetakis and Cook (2002) [1] introduced a comprehensive framework for genre classification based on audio features. Following this, several researchers have investigated the use of deep learning techniques, such as convolutional neural networks, to improve the classification of audio samples [2]. In recent years, researchers have also considered the potential of natural language processing models in genre classification. For instance, Mayer et al. (2008) [3] and Fell and Sporleder (2014) [4] explored the use of lyrics to distinguish between music genres. Multi modal approaches for this task found in the literature combine audio and song lyrics as text. Some studies that explore these approaches are by Laurier et al. (2008) [5] and Neumayer et al. (2007) [6]. The existing body of research in this domain highlights the importance of music genre classification and its potential to enhance the music listening experience. By building upon these findings, our project seeks to explore and compare multiple approaches.

5 Dataset and Features

The data set contains 5 music genres - country, metal, rock, pop, and rap. The dataset was compiled by querying the Spotify API for a genre, retrieving all associated artists within that genre, and subsequently obtaining all tracks and their audio features (danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence, and tempo) as well as the 30-second mp3 previews. Initially, the dataset featured an imbalanced distribution of tracks across genres, with 1767 country tracks, 1271 metal tracks, 1263 rock tracks, 1058 pop tracks, and 982 rap tracks. To create a balanced dataset, we randomly sampled 982 tracks from each genre. The data set was created using Spotify API and LastFM for lyrics.

The data set has been split into a training set with 3,928 samples and a test set with 982 samples using a stratified 80:20 train-test split. This data set can be used to train machine learning models for genre classification tasks.

Spotify API provides different feature extraction paths one of which is the meta data features. These include the features like Danceability that describes how suitable a track is for dancing based on elements such as tempo, rhythm stability, and overall regularity. Acousticness, which is a measure of whether a track is acoustic or not. Energy is a measure of intensity and activity in a track. Instrumentalness is a measure of whether a track contains vocals or not. Liveness detects the presence of an audience in the recording, and loudness is the overall loudness of a track. Speechiness detects the presence of spoken words in a track. Tempo is the overall estimated tempo of a track in beats per minute (BPM), and valence is a measure of the musical positiveness conveyed by a track.

In order to understand the influence of metadata features in genre classification we created the Box plot of every feature for each genre, the heat map of of the features and the T-SNE plot of the data set.

Based on the analysis of the box plots for each feature and genre, we can observe the following:

- Danceability: Rap and pop songs tend to have higher danceability scores than country and metal songs.

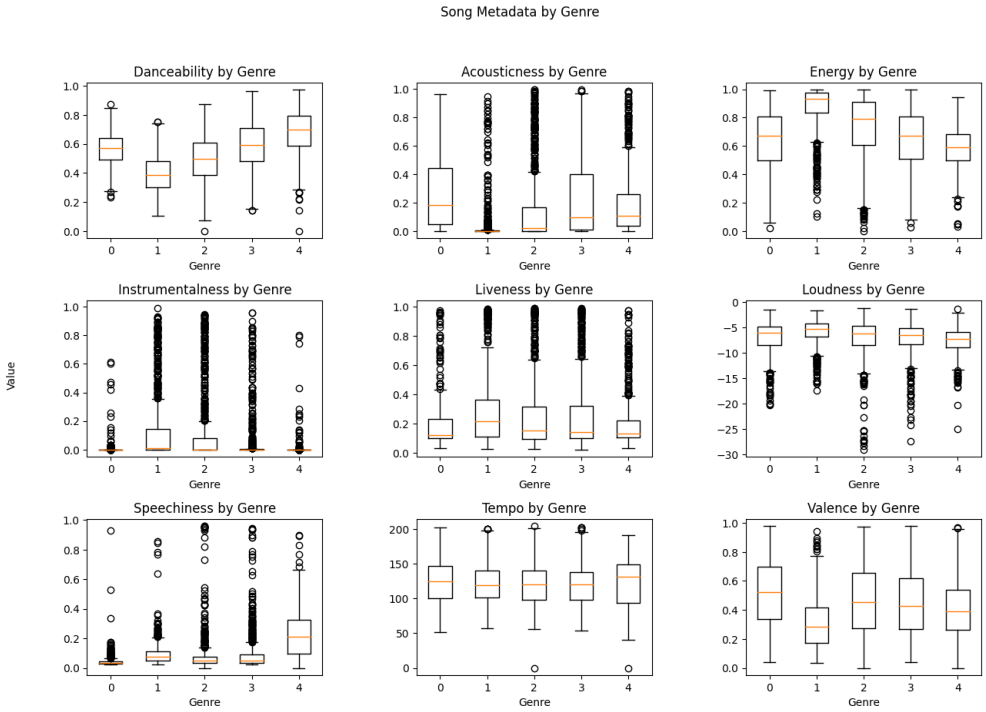

Figure 2: Boxplot of every feature with each genre | 0 - Country, 1 - Metal, 2- Rock, 3 - Pop, 4 - Rap

- Acousticness: Country and pop songs tend to have higher acousticness scores than rock, metal and rap songs. The range of acousticness is higher for country and pop songs.
- Energy: Metal and rock songs tend to have higher energy scores than country, pop and rap songs. The range of energy is higher for country, rock and pop songs.
- Instrumentalness: The median instrumentalness score is similar for all genres.
- Liveness: Metal and rock songs tend to have higher liveness scores than country, pop and rap songs. The range of liveness is similar for all genres.
- Loudness: The median loudness score is similar for all genres.
- Speechiness: Rap songs tend to have higher speechiness scores than other genres. The median speechiness score is similar for other genres.
- Tempo: The median tempo score is similar for all genres. The range of tempo is higher for rap songs.
- Valence: Country and pop songs tend to have higher valence scores than rock, metal and rap songs. The range of valence is higher for country, rock and pop songs.

Overall, we can observe that each genre tends to have unique characteristics in terms of the metadata features analyzed. For example, country songs tend to have higher acousticness and valence scores,

while rap songs tend to have higher speechiness and tempo ranges. Metal and rock songs tend to have higher energy and liveness scores. Pop songs generally fall in the middle for most features, while still showing some variation in range.

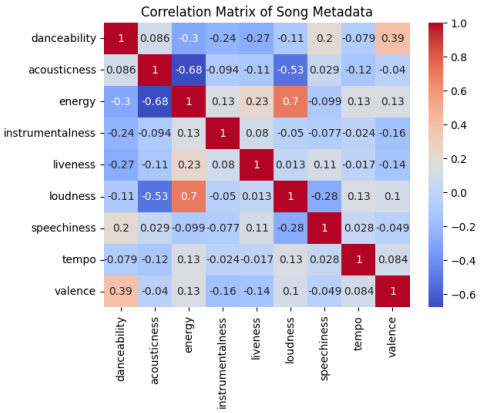

Figure 3: Correlation matrix of features

To visualize our results we have plotted a heat map that describes the correlation between different metadata features for the different music genres. Each cell in the heatmap corresponds to the correlation coefficient between two features. This technique can be useful for identifying relationships between different features and understanding how they are related to each other in each genre. For example, if two features have a high positive correlation for a particular genre, this suggests that when one feature is high, the other feature is also likely to be high. Similarly, if two features have a high negative correlation, this suggests that when one feature is high, the other feature is likely to be low. By analyzing the heatmap, we can gain insights into how different features are related to each other within each genre and how they contribute to the overall sound and feel of the music.

From the heatmap, we can see that there is a strong negative correlation between acousticness and energy and loundness. This makes sense as acoustic songs tend to be quieter and more subdued, while energetic songs are louder and more intense. On the other hand, there is a strong positive correlation between energy and loudness, which suggests that songs with higher energy tend to be louder as well. There is also a moderate positive correlation between valence and danceability indicating that more positive or happy songs tend to have higher dance levels. Finally, there is a moderate positive correlation between energy and liveness.Overall, the heatmap allows us to visualize the relationships between the different features, and identify patterns and trends in the data.

A T-SNE plot can provide insights into the structure of the data and the relationships between different features. In this plot, each dot represents a song in the metadata, and the color of the dot represents the genre of the song. From the the TSNE plot we observe that there isn't a clear separation of the different genres into distinct clusters. Instead, the dots are scattered randomly throughout the plot. This could indicate several things. It's possible that the metadata features we are using are not strong predictors of genre and don't provide clear differentiation between different genres. It's also possible that the T-SNE algorithm wasn't able to find clear patterns in the data, or that the dataset is too small or too diverse to identify clear clusters.

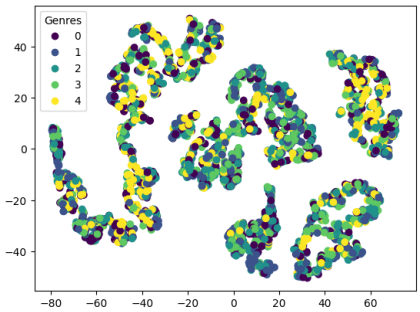

Figure 4: TSNE plot of train data

It's important to keep in mind that music genres are often subjective and can overlap or change over time, so it's possible that there isn't a clear separation between different genres in the data.

6 Experiments

6.1 Classification using Spotify's meta features

We explored a comprehensive set of classifiers with the preprocessed data , fine-tuning each classifier using grid search and cross-validation. Finally, we evaluated the performance of each classifier on the test data, analyzing metrics such as accuracy, F1 score, and confusion matrices

6.1.1 Models for classification using meta features

- Logistic Regression: A statistical method used to analyze a dataset with one or more independent variables that determine an outcome. It is often used in binary classification problems, and is simple to implement and interpret.
- K-Nearest Neighbors: A non-parametric method that is used classification and regression problems. It is based on the idea that similar data points are likely to have similar labels, and it determines the label of a new point based on the k-nearest neighbors in the training data.
- Support Vector Machines: A classification method which separates data points into classes using a hyperplane that maximizes the margin between the classes. It is effective in high-dimensional spaces and can handle nonlinear relationships between data points.
- Decision Trees: A method which creates a model of decisions and their possible consequences. It is easy to interpret and can handle both categorical and numerical data.
- Random Forest: A classification and regression methods which combines multiple decision trees to improve accuracy and reduce overfitting. It is able to handle missing values and noisy data, and can handle both categorical and numerical data.
- Gradient Boosting [7]: A method which combines multiple weak models to create a strong model. It is able to handle missing values and noisy data, and can handle both categorical and numerical data.

- Extra Trees [8]: are used for classification and regression problems, which is similar to Random Forest but uses a different method for selecting the splits at each node. It can handle both categorical and numerical data, and is less prone to overfitting than Random Forest.
- AdaBoost [9]: Combines multiple weak models to create a strong model. It is able to handle both categorical and numerical data, and is effective in reducing bias and variance.
- XGBoost [10]: A method which is similar to Gradient Boosting but uses a different method for selecting the splits at each node. It is able to handle missing values and noisy data, and can handle both categorical and numerical data.
- MLPClassifier: A neural network-based method used for classification problems, which uses multiple layers of nodes to learn complex relationships between the input and output. It is able to handle both categorical and numerical data, and is effective in learning non-linear relationships.

In **logistic regression**, we experimented with hyperparameters such as regularization strength and maximum number of iterations for convergence. By performing a grid search, we found that the best hyperparameters were C=10 and max_iter=1000, indicating that a stronger regularization was preferred over a higher number of iterations for this dataset.

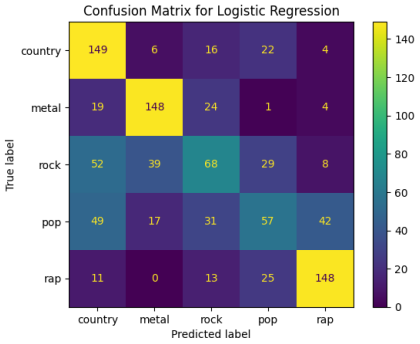

Figure 5: Confusion Matrix for Logistic Regression

The **K-Nearest Neighbors**(KNN) algorithm uses hyperparameters to control the distance metric, number of neighbors, and weighting scheme used to combine labels. We searched over the hyperparameter "n neighbors" and found that 10 worked best for our dataset. We also tried different distance metrics and weighting schemes, but found that the default settings worked best.

Support Vector Machines(SVM) algorithm is a classification technique that finds a hyperplane to separate data points of different classes with maximum margin. The SVM algorithm has hyper parameters that control the trade-off between maximizing the margin and minimizing the classification error, as well as the choice of kernel function and its parameters.

To find the best hyper parameters for our data set, we performed a grid search over the "C" and "gamma" hyper parameters. We found that the best values were C=10 and gamma=0.1, indicating a strong regularization strength and a reasonable width of the kernel.

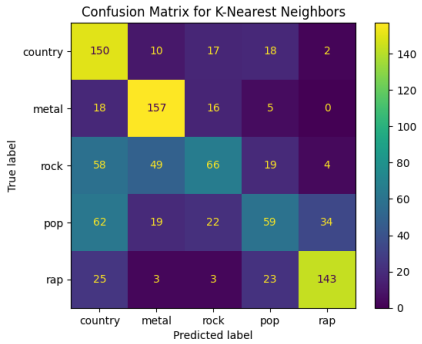

Figure 6: Confusion Matrix for KNN

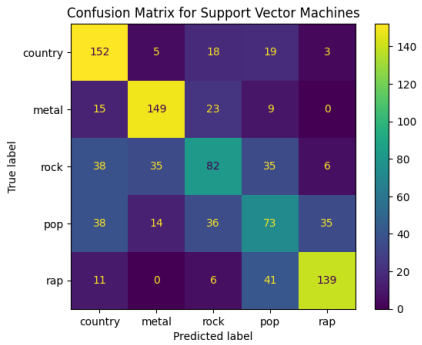

Figure 7: Confusion Matrix for SVM

Decision Trees are a classification algorithm that partitions the feature space based on input features. Hyper parameters control the depth of the tree, impurity measure, and stopping criteria for leaf nodes. In our experiment, we performed grid search on the "max_depth" hyper parameter and found the best value to be 8.

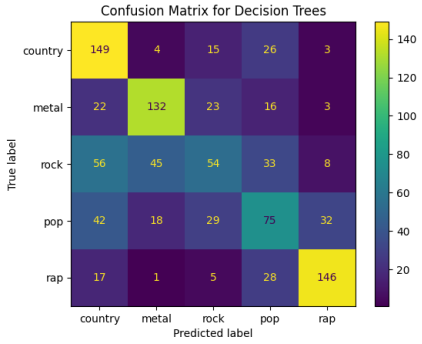

Figure 8: Confusion Matrix for Decision Tree

In **Random Forest**,we conducted experiments on two hyper parameters, the number of decision trees (n_estimators) and the maximum depth of each tree (max_depth), for Random Forest. By performing a grid search, we determined that the optimal configuration for our data set was a forest consisting of 100 trees, where each tree has a maximum depth of 10.

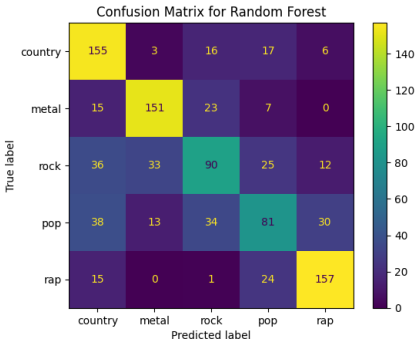

Figure 9: Confusion Matrix for Random Forest

For **Gradient Boosting**,The hyper parameters of number of estimators, learning rate, and maximum depth of the tree were experimented with for the Gradient Boosting model. The best combination was found to be n_estimators=100, learning_rate=0.1, and max_depth=5, indicating that a moderate number of estimators and learning rate with a relatively deep tree is optimal for this data set.

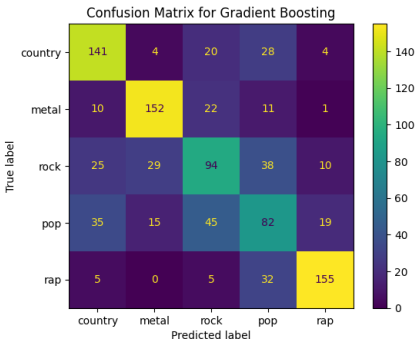

Figure 10: Confusion Matrix for Gradient Boosting

For **Extra Trees**, we experimented with hyperparameters related to the number and depth of decision trees in the Extra Trees ensemble for genre classification.We found that the best hyperparameters were 150 trees with a maximum depth of 10.

For **AdaBoost**,the control the number of weak learners and the learning rate. We found the best values for these hyperparameters to be 150 for the number of estimators and 0.1 for the learning rate, using the SAMME.R algorithm.

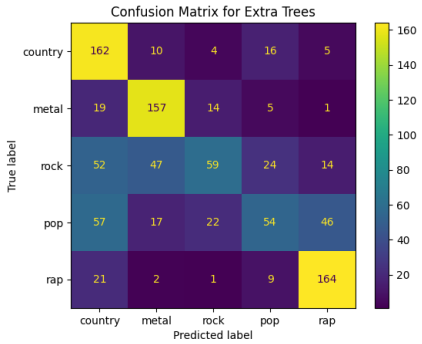

Figure 11: Confusion Matrix for Extra Trees

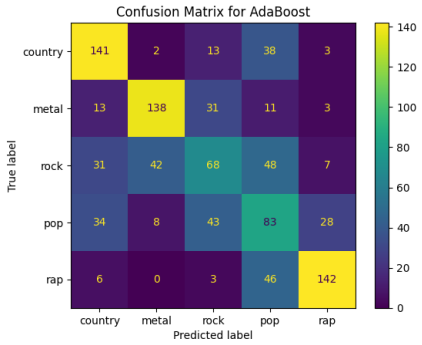

Figure 12: Confusion Matrix for Adaboost

For **XGBoost** In our XGBoost model, we experimented with the hyperparameters "n_estimators", "learning_rate", and "max_depth" using grid search. We found that the best combination of hyperparameters was when "n_estimators" was set to 150, "learning_rate" was set to 0.1, and "max_depth" was set to 5. These hyperparameters produced the best classification results on the dataset.

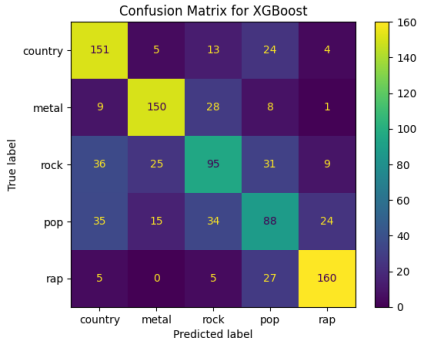

Figure 13: Confusion Matrix for XGBoost

For **MLPClassifier**, The best hyperparameters for MLPClassifier were found using grid search, with the activation function set to "tanh", the hidden layer size set to (100,), and the optimization solver set to "adam". The model achieved the best classification performance with these hyperparameters.

For all the models that we tried, we have obtained a confusion matrix for the best hyperparameters, which shows the classification performance of the respective model. The yellow color indicates that the genre is classified accurately. A color bar is shown in order to navigate the accuracy of the genres classified by respective models.

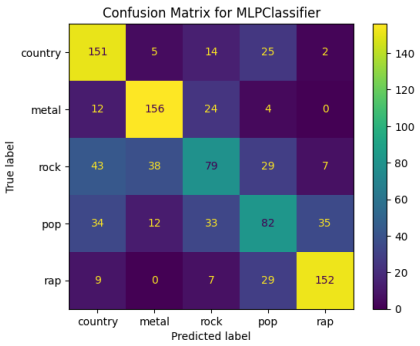

Figure 14: Confusion Matrix for MLPClassifier

6.2 Classification using audio features extracted from MP3 samples

In addition to using audio features for classification, we also explored genre classification using features extracted directly from the 30-second mp3 samples. We employed the Librosa library to extract Mel-frequency cepstral coefficients (MFCCs), chroma features, spectral contrast, and tonnetz, resulting in a feature vector of size 74 for each sample. We then utilized Random Forest, SVM, and a custom fully connected feedforward neural network to classify the genres.

The custom neural network architecture consisted of the following layers:

1. Input layer with size input_size.
2. Fully connected (Linear) layer with 256 units followed by Batch Normalization, ReLU activation, and 0.5 Dropout.
3. Fully connected (Linear) layer with 64 units followed by Batch Normalization, ReLU activation, and 0.5 Dropout.
4. Fully connected (Linear) layer with 64 units followed by Batch Normalization, ReLU activation, and 0.5 Dropout.
5. Fully connected (Linear) layer with 64 units followed by Batch Normalization, ReLU activation, and 0.5 Dropout.
6. Fully connected (Linear) layer with 32 units followed by Batch Normalization, ReLU activation, and 0.5 Dropout.
7. Output layer, a fully connected (Linear) layer with num_classes units.

The classification process for the features extracted from mp3 files followed steps similar to those carried out using audio features. The different approaches yielded the following confusion matrices.

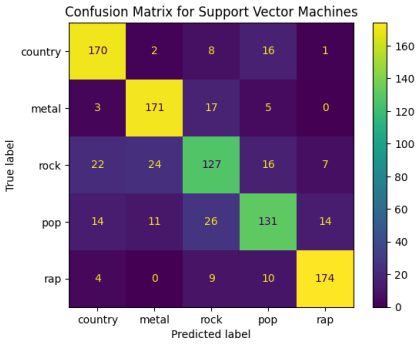

Figure 15: Confusion Matrix for SVM

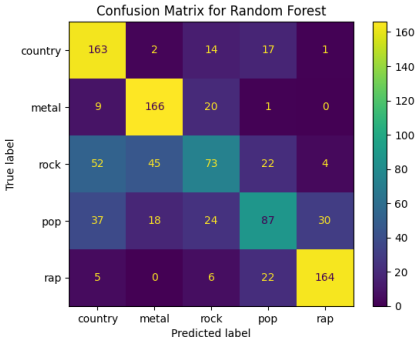

Figure 16: Confusion Matrix for Random Forest

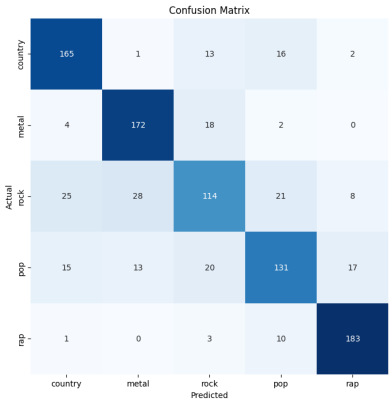

Figure 17: Confusion Matrix for custom neural network

6.3 Classification using lyrics

For genre classification the lyrics of a song are first converted into a matrix of word frequencies using a CountVectorizer function. This matrix is then converted into a tensor and fed into a neural network model for training.

We started out with a multinomial Naive Bayes since it is computationally efficient and relatively simple to implement. It also works well with sparse data and has been shown to be effective even with relatively small amounts of training data. However we noted that the accuracy was very low.

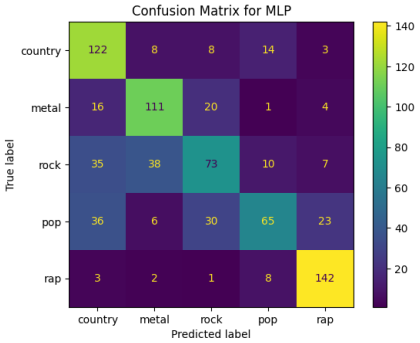

Figure 18: Confusion Matrix for Multinomial Naive Bayes

Then we developed a simple neural network model with two fully connected layers. The first layer is the input layer, which takes the tensor of word frequencies as input. The second layer is the output layer, which produces the probability distribution over the different genres. The activation function used in all layers was Leaky ReLU, which helps to mitigate the vanishing gradient problem during training.

To train the model, we used batch stochastic gradient descent with a batch size of 32. This means that the model updates its parameters based on the average of the gradients computed over a subset of the training data. The optimizer used was Adam, which is a popular optimization algorithm that adapts the learning rate for each parameter based on its gradient history. The loss function used was cross-entropy loss, which measures the difference between the predicted probability distribution and the actual distribution of the genre labels.

The combination of learning rate and momentum that gave the best accuracy of 0.58 was a learning rate of 0.01 and a momentum of 0.9. The learning rate controls how much the parameters of the model are updated during each iteration of the optimization algorithm. A higher learning rate can lead to faster convergence but may also cause the optimization to overshoot the optimal parameters. The momentum parameter helps to smooth out the update process by incorporating a fraction of the previous update direction into the current update direction. A higher momentum can help to accelerate the optimization process and overcome local minima.

Overall, our approach was a good starting point for genre classification using lyrics. We further developed Recurrent Neural Network (RNN) model. RNNs are commonly used in natural language processing tasks because they can process sequential input data such as text.

Our RNN model has an input layer that takes the sequence of word embeddings, a hidden layer with RNN cells that process the sequential information, and a fully connected linear output layer that produces the predicted genre probability distribution. The RNN layer allows the model to capture the contextual relationships between the words in the lyrics.

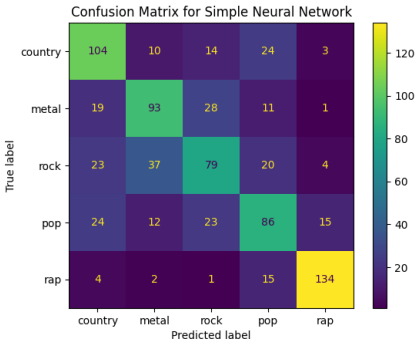

Figure 19: Confusion Matrix for Simple Neural Network

We tried different hyperparameters to optimize the model's performance. One important hyperparameter that we tuned is the learning rate. From our experiments, we found that a learning rate of 0.02 gave the best accuracy for this task.

We trained the RNN model for 50 epochs, which means that the model went through the entire training dataset 50 times during the training process. By training the model for multiple epochs, the model can learn to adjust its parameters to minimize the training loss and improve its classification accuracy.

Overall, the RNN model is a good choice for text classification tasks, and our experiments with hyperparameter tuning and training epochs can help optimize the model's performance for this specific task. However, it's important to note that there are other factors to consider, such as the choice of RNN architecture (e.g., LSTM or GRU), the choice of activation functions, and the size and number of hidden layers.

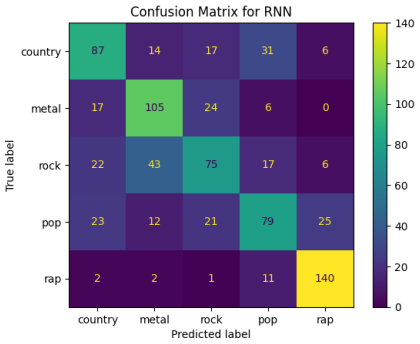

Figure 20: Confusion Matrix for RNN

We also experimented with replacing the RNN layer in our model with a Gated Recurrent Unit (GRU) layer. GRU is a type of RNN that is designed to be better at capturing long-term dependencies in sequential data by selectively remembering or forgetting information.

Compared to RNN, GRU has fewer parameters and is easier to train, making it a popular choice in many natural language processing tasks.

By replacing the RNN layer with a GRU layer in our model, you were able to improve the accuracy of the model. This could be due to the GRU's ability to selectively store or discard information from

previous time steps, which helps the model better capture the underlying patterns in the lyrics.

In addition to the change in the type of layer used, we trained the GRU model similarly to the RNN model by tuning the same hyperparameters such as the learning rate and number of epochs. This allowed us to make a fair comparison between the two models.

Overall, the GRU layer is a good choice for text classification tasks due to its ability to selectively remember or forget information from previous time steps.

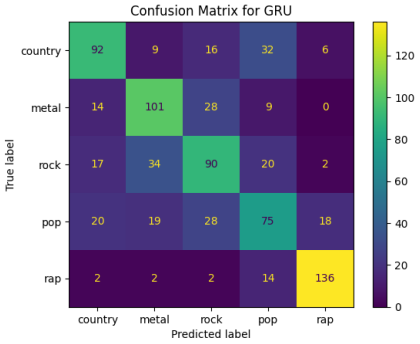

Figure 21: Confusion Matrix for GRU

We also implemented BERT, which is a transformer-based model that has achieved state-of-the-art performance on many natural language processing tasks. We trained the BERT model for three epochs and achieved a training loss of 0.130 for one epoch. This means that the model was able to learn from the training data and adjust its parameters to minimize the difference between the predicted genre probabilities and the true genre labels. However, we did not have sufficient memory to run all three epochs, which may limit the performance of the model.

7 Results and Conclusion

Model	Accuracy	F1 Score
Random Forest	0.6649	0.6507
SVM	0.7871	0.7851
Custom fully connected feed forward neural network	0.7790	0.7746

Table 1: Performance on MP3 Features

The aim of this project was to explore and compare different methods for music genre classification using a dataset provided by the Spotify API. In order to achieve this goal, three distinct approaches were examined, including analyzing 30-second MP3 samples of tracks, examining the lyrical content of tracks, and utilizing audio features of tracks retrieved through Spotify's API. The performance of the models generated by each approach was evaluated using accuracy and F1-score metrics.

Analysis of 30-second MP3 samples of tracks yielded the best performing model, which was an SVM model. This model achieved an accuracy score of 0.7872 and an F1-score of 0.7852. The utilization

Model	Accuracy	F1 Score
Logistic Regression	0.5406	0.5257
K-Nearest Neighbors	0.5527	0.5344
Support Vector Machines	0.6078	0.5930
Decision Trees	0.4794	0.4838
Random Forest	0.6018	0.5952
Gradient Boosting	0.6088	0.5988
Extra Trees	0.5838	0.5736
AdaBoost	0.5507	0.5184
XGBoost	0.5577	0.5763
MLPClassifier	0.6078	0.5452

Table 2: Performance on Audio Features

Model	Accuracy
Multinomial Naive Bayes	0.56
Simple Neural Network	0.58
Recurrent Neural Network	0.68
RNN with gated recurrent unit	0.687

Table 3: Performance on Lyrics

of audio features of tracks retrieved through Spotify's API resulted in a gradient boosting model, which achieved an accuracy score of 0.6088 and an F1-score of 0.5988. Finally, the examination of the lyrical content of tracks produced a model based on the gated recurrent unit (GRU) algorithm, which achieved an accuracy score of 0.68.

The results of this study demonstrate that analyzing 30-second MP3 samples of tracks is the most effective method for music genre classification. This approach outperformed the other two methods and achieved the highest accuracy and F1-score metrics. However, it is important to note that the utilization of audio features through Spotify's API also produced a relatively accurate model, albeit with lower accuracy and F1-score than the MP3 analysis method. Meanwhile, the examination of lyrical content resulted in a lower accuracy model.

In conclusion, this research project provides valuable insights into the various methods that can be used for music genre classification. The findings suggest that analyzing 30-second MP3 samples of tracks is the most effective approach, although the utilization of audio features through Spotify's API may also yield relatively accurate results. These results have important implications for the development of more sophisticated music recommendation systems and other related applications in the music industry.

References

[1] G. P.Cook, "Musical genre classification of audio signals | ieee journals ..." 2002. [Online]. Available: https://ieeexplore.ieee.org/document/1021072

[2] K. Choi, "Automatic tagging using deep convolutional neural networks," Jun 2016. [Online]. Available: https://doi.org/10.48550/arXiv.1606.00298

[3] S. Bai, V. Koltun, and Z. J. Kolter, "Stabilizing equilibrium models by jacobian regularization," Jun 2021. [Online]. Available: https://arxiv.org/abs/2106.14342

[4] R. Mayer, R. Neumayer, and A. Rauber, "Rhyme and style features for musical genre classification by song lyrics," 2008. [Online]. Available: https://publik.tuwien.ac.at/files/PubDat$_1$66272.*pdf*

[5] C. Laurier, "Multimodal music mood classification using audio and lyrics | ieee ..." [Online]. Available: https://ieeexplore.ieee.org/document/4725050/

[6] R. Neumayer and A. Rauber, "Integration of text and audio features for genre classification in music information retrieval," Jan 1970. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-540-71496-5$_7$8

[7] J. H. Friedman, "Greedy function approximation: A gradient boosting machine." [Online]. Available: https://projecteuclid.org/journals/annals-of-statistics/volume-29/issue-5/Greedy-function-approximation-A-gradient-boosting-machine/10.1214/aos/1013203451.full?tab=ArticleLink

[8] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees - machine learning," Mar 2006. [Online]. Available: https://link.springer.com/article/10.1007/s10994-006-6226-1

[9] Y. Freund and R. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," Dec 1996. [Online]. Available: https://www.face-rec.org/algorithms/Boosting-Ensemble/decision-theoretic$_g$*eneralization.pdf*

[10] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," Jun 2016. [Online]. Available: https://arxiv.org/abs/1603.02754