DS 4400: Machine Learning and Data Mining I

Spring 2024

Genre Classification for Musical Snippets

Devyanshi Chandra, Jake Phelan, Veronica Yang

Link to Code **Link to Video**

Problem Description

Throughout this course, we have explored the several directions machine learning has progressed in its ability to adapt and predict various types of data, ranging from industry-specific computer science issues to commonly utilized algorithms on social media. For this project, we intended to explore the usage of audio and test various models' capabilities in identifying a music snippet's genre. This project aims to replicate how music streaming services such as Spotify and Apple Music recommend user's music through curated playlists. Although the genre is a provided data point to them, often these services go beyond just the information given to them regarding each song and provide more personalized insights and curated playlists based on analyzing specific songs. They frequently employ sophisticated algorithms to analyze specific characteristics of each song, enabling them to offer users more tailored insights and recommendations based on their preferences and listening habits. Additionally, this provides us with an opportunity to explore the nuances of audio data analysis, including feature extraction, model selection, and evaluation metrics. This ultimately contributes to our understanding of how machine learning can be applied to tasks such as music genre classification and recommendation systems.

Dataset and Exploratory Data Analysis

The GTZAN Music dataset is a commonly used dataset available on Kaggle, in which users have created a multitude of projects related to this subject. This inspired us to use the file, as it covers a diverse range of music genres with over 100 audio clips. These snippets of the songs have been split up into 3-second intervals, which was used in our models due to the vast amount of data available to us compared to the other CSVs. We did not have the opportunity to incorporate images into this project due to time and complexity constraints. For each audio clip, a set of features is extracted to represent its acoustic characteristics. These features include spectral features such as Mel-frequency cepstral coefficients (MFCCs), and rhythmic features such as tempo and beat information. The dataset is labeled with the corresponding genre for each audio clip, including "classical", "disco", "jazz", "pop", "rock", "blues", "country", "hiphop", "metal", and "reggae".

Before attempting to create our models, we went forward with some data preprocessing and exploratory data analysis (EDA) to allow ourselves to gain insights into the structure, distribution, and relationships within the dataset. In the context of the GTZAN Dataset for Music Genre Classification, EDA involves a systematic examination of the dataset's features and their potential impact on genre classification.

Firstly, as with any machine learning task, it was important to preprocess our data: we sought to partition our data into testing and training sets as well as to encode and normalize all samples. We read the CSV file into a Pandas DataFrame, dropping the file name feature as it would convolute our models' abilities to blindly predict a snippet's genre. We also dropped the snippet length feature, as it took on a singular value across all samples. Next, the feature columns were extracted into a single "X" variable. The label column ("genre") was in a string format, so it was crucial to encode it. We implemented one-hot encoding, which was deemed more appropriate than an ordinal encoding, as there is no inherent ordering to the genres. The encoding resulted in 10 binary label columns which were then casted to a NumPy array and stored in a single "y" variable. Finally, using scikit-learn's train-test split method, our data was partitioned into X_train, X_test, y_train, and y_test sets with a random state of 42 for reproducibility and a test size of 20%. Finally, we fit a scikit-learn StandardScaler to the X_train data and used it to transform both the X_train and X_test data to have a zero mean and unit variance.

We started our analysis by printing DataFrame information to examine the completeness and structure of our data, specifically checking to ensure that there were no null entries. Once we identified no missing values, we moved on to the descriptive statistics of our datasets (the complete set and training and testing sets). These statistics included the mean, standard deviation, minimum, and maximum values of each feature. This process helps provide a basic understanding of the data for further, more complex analysis.

After reading the data, we created a Correlation Heat Map, providing us with a visual interpretation of each feature's relationship with the other.

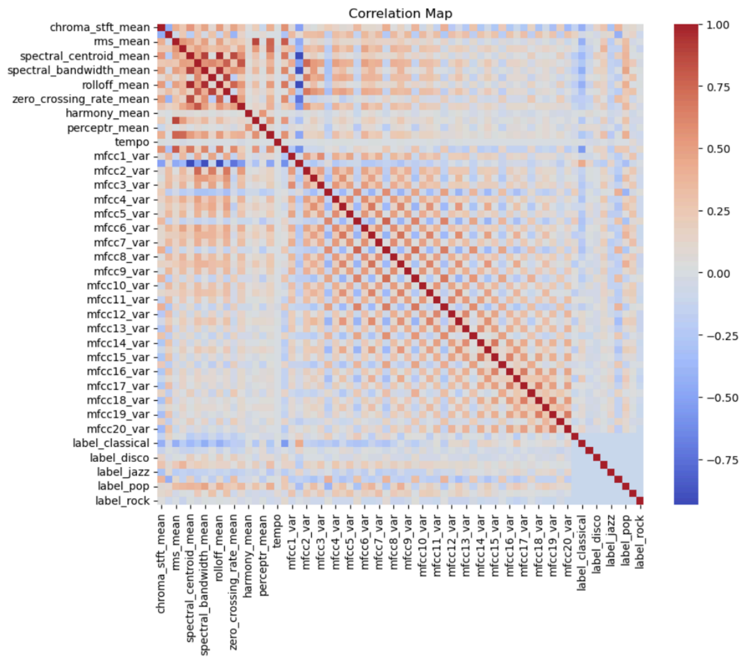


Figure 1: *Correlation heat map with gradient portraying the correlational relationship*

While the correlation between features and genre labels appears relatively weak, some patterns can be observed. For example, the feature "tempo" and the label "classical" have a darker blue color, indicating a negative correlation. Tempo is defined as the speed of the music or the beats per minute. Classical music encompasses a wide variety of tempos, ranging from being slower to faster paced. However, if we look at classical music from a more generalized lens, it tends to have a slower, more deliberate tempo when compared to genres like "pop" or "rock". We also observed some slight patterns between features. Features that share similar musical properties have a positive correlation. Spectral features encompass all information on the frequency of the sample, so features that fall under this category demonstrate a positive correlation.

Approach and Methodology

Initially, we intended to incorporate Convolutional Neural Networks (CNN) in depth with the audio snippets available to us, however with the nature of the data and the lack of volume required to move forward with the idea, we chose to work with support vector machine (SVM) and k-nearest neighbors (KNN) models. Additionally, we intended to implement Linear Discriminant Analysis (LDA) as a part of feature selection. Using SVM and KNN, along with GridSearchCV for parameter tuning, was the pragmatic choice on our end for classifying music genres. SVM and KNN algorithms provide effective and efficient classification and are capable of handling both numerical and categorical data. With the GTZAN Dataset, which comprises audio excerpts, these algorithms can utilize the extracted features to discern patterns characteristic of different music genres. Additionally, GridSearchCV optimized the hyperparameters of the KNN model, enhancing its performance further. Although CNNs are powerful for analyzing sequential data like audio, their efficacy depends heavily on large volumes of data for training, which may not have been available in this context. As for LDA for feature selection, its application might have been challenging due to the high dimensionality of audio feature spaces and the complex relationships between features and genre labels. LDA typically assumes linear separability between classes, which may not hold for the diverse and intricate nature of music genres represented in the dataset. Hence, prioritizing SVM and KNN alongside GridSearchCV allowed for a more practical and effective approach to genre classification, given the characteristics of the dataset and the objectives of the project.

XGBoost

XGBoost, short for Extreme Gradient Boosting, is an algorithm known for its efficiency and accuracy in solving classification problems. It combines the strengths of gradient boosting techniques with regularization to optimize performance. However, by default, it is suited much better for binary classification problems than for multiclass classification. Therefore for this model, we first initialized XGBoost as a binary classifier with a logistic objective loss function. We then used this model as the base classifier in a scikit-learn multi-output classifier model. This model is trained on our training dataset with corresponding genre labels. Then we predict our testing data using the trained model and evaluate its performance by computing the accuracy score and other metrics.

AdaBoost

After implementing XGBoost, we were interested in comparing its performance with that of another boosting method. AdaBoost, or Adaptive Boosting, is a simple ensemble learning method which combines weaker learners to create a strong classifier. By adjusting the weights of instances at each iteration according to their misclassification status, AdaBoost focuses on correctly classifying difficult samples. For our AdaBoost model, we used a simple decision tree as the base classifier. These base classifiers each had a maximum depth parameter of 1 to avoid overcomplexity at the ensemble level. Once the base classifier was fed to the AdaBoost model, the ensemble was trained to our data. We then predicted the labels of the testing data and computed accuracy and other performance metrics.

KNN

We then implement KNN within a pipeline for enhanced efficiency and performance. To address class imbalance—a common issue in classification tasks, especially with larger datasets that have imbalanced data within labels—we incorporated Synthetic Minority Over-sampling Technique (SMOTE) to oversample the minority classes during training. We also incorporated Principal Component Analysis (PCA) to reduce the dimensionality of our feature space while retaining 95% of the variance, which enhanced efficiency and mitigated dimensionality. We specify a parameter grid for KNN, encompassing different configurations for the number of neighbors, weight schemes, and distance metrics. For each parameter, we tested between 3, 5, and 7 numbers of neighbors, between uniform and distance as the weight metric, and between Manhattan and Euclidean distance for distance metrics.

Following, we performed a grid search with 5-fold cross-validation to identify the optimal hyperparameters that maximize the model's performance. We obtained the best-performing model from the search results. The optimal number of neighbors was found to be 3, using Euclidean distance for distance metrics and distance as the weight metric. These parameters were implemented on the model and created a comprehensive classification report, providing insights into the precision, recall, and F1-score for each genre class, along with macro and weighted averages across all classes. We also handled cases where the division by zero may occur, ensuring robustness in our evaluation metrics. This systematic approach enables us to fine-tune our classification model and gain deeper insights into its effectiveness in genre classification tasks.

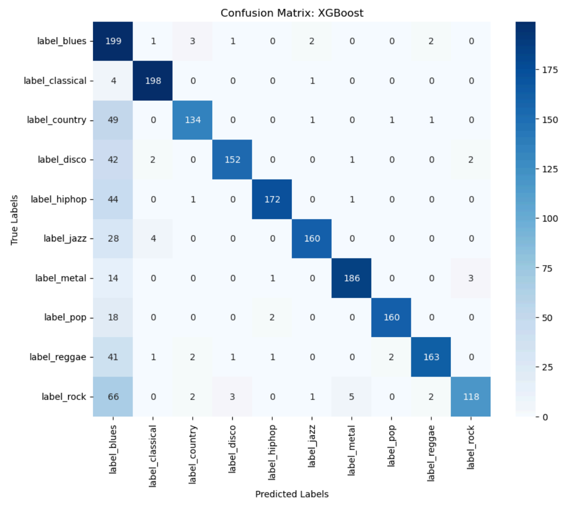
<u>SVM</u>

We chose to implement an SVM model as it was well suited to the requirements of our problem space and the available data. SVMs are ideal for highly-dimensional data and multiclass classifications, which made it a perfect candidate for our genre classification. This model was very self-contained, like XGBoost, and did not require the extra specifications of KNN. Using a linear kernel, we were able to instantiate and train an SVM model from scikit-learn. We chose to use a ridge penalty of coefficient 1 for regularization. Like before, we finish up the testing of this model by predicting our testing data and reporting the accuracy score and other performance metrics.

Discussion and Result Interpretation

After experimenting with the various aforementioned machine learning algorithms, we obtained the following results:

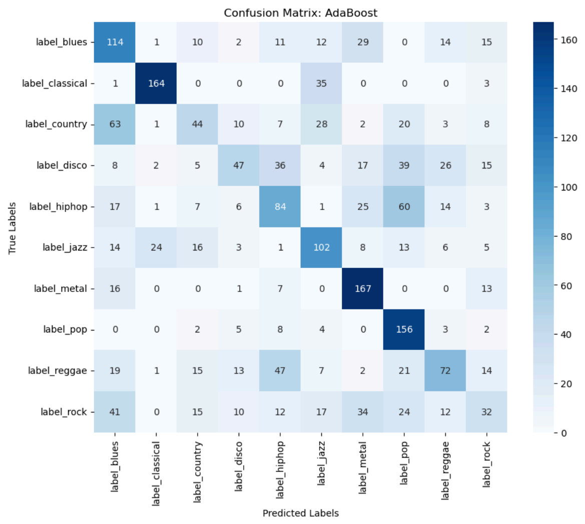
<u>XGBoost</u>



XGBoost achieved an accuracy of 0.7932 on the test dataset. The weighted average precision, recall, and F1-score were 0.96, 0.81, and 0.87 respectively. An average precision score of 0.96 implies that XGBoost was very effective at correctly identifying the genre of most musical snippets. In fact, this is visible in the model's confusion matrix. The majority of the weight is along the matrix diagonal, meaning that many predictions were correct. Inspecting cells in the upper triangular area and lower triangular area, most instances of incorrect predictions have a multiplicity of 0, 1, 2, or occasionally 3. The first column, corresponding to samples that XGBoost predicted to be "blues" songs, has an uncharacteristic amount of incorrect prediction. However, there are still 199 correctly-predicted “blues” snippets”—higher than any other category in the matrix. It seems that “blues” snippets are the most populous in the data, and they may have a higher variability among them compared to snippets of other genres. Overall, XGBoost was a successful estimator of genre, even with some room for improvement. Despite its performance, it's important to again mention that XGBoost is primarily designed for binary classification, and its general efficacy for multiclass classification may be limited.

AdaBoost

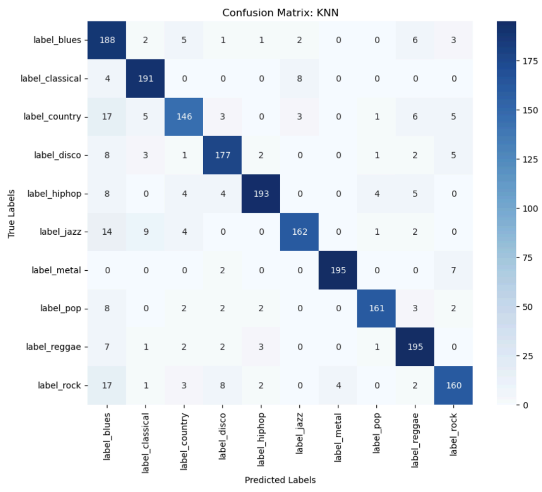
AdaBoost yielded an accuracy of 0.4914 on the test dataset. While this accuracy indicates relatively poor performance, further insights can be gleaned from its other metrics. The weighted average precision was 0.48; the recall was 0.49; and the F1-score was 0.47. An average precision score of 0.48 suggests that AdaBoost struggled to correctly identify the genre of musical snippets. This observation is reinforced by the confusion matrix, where only certain genres ("blues", "classical", "metal", and "pop") exhibit higher accuracy, while the remaining genres are often wrongly-predicted. Overall, AdaBoost's performance highlights its limitations in effectively classifying music genres, with only a subset of genres demonstrating satisfactory accuracy. This suggests that a decision tree-based AdaBoost model may not be the most suitable choice for this particular classification task.



Boost Comparison

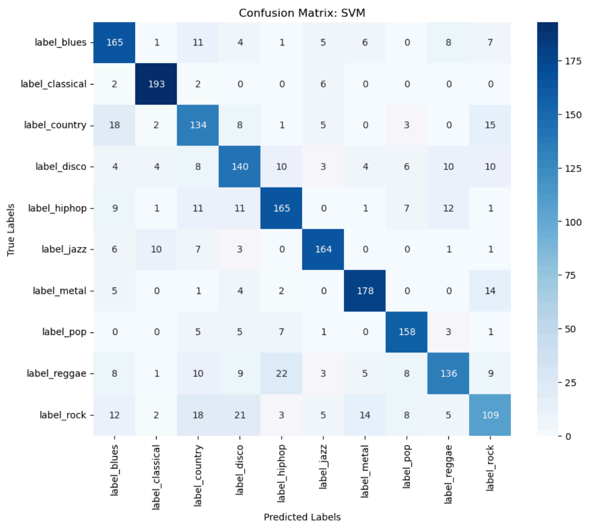
When comparing XGBoost and AdaBoost for classification, it is important to consider their underlying boosting strategies. XGBoost is a bit more sophisticated and leverages gradient boosting with regularization to enhance model performance. In contrast, AdaBoost, with its adaptive boosting approach, focuses on iteratively adjusting the weights of misclassified instances to improve model accuracy. In this project, XGBoost significantly outperformed AdaBoost, as shown by its higher accuracy and precision scores. XGBoost's ability to effectively handle the complexity of the dataset and discern patterns across multiple classes contributed to its superior performance. Overall, the comparative performance of XGBoost and AdaBoost underscores the importance of selecting the appropriate boosting strategy for the specific characteristics of the dataset and classification problem at hand. While AdaBoost may excel in certain scenarios, XGBoost's more advanced boosting techniques make it better suited for complex classification tasks like music genre classification, as demonstrated in this project.

<u>KNN</u>



KNN demonstrated strong performance with an accuracy of 0.8769 on the test dataset, showcasing its effectiveness in classifying music snippets into different genres. Additionally, the weighted average precision, recall, and F1-score were 0.92, 0.88, and 0.90 respectively, indicating high levels of consistency in predictions across all genres. The KNN algorithm, combined with preprocessing techniques such as SMOTE and PCA, proved to be instrumental in addressing key challenges encountered in the dataset. The utilization of SMOTE helped alleviate class imbalance issues, ensuring that the model was not biased towards genres with more samples, like "blues". Additionally, PCA enabled dimensionality reduction while retaining 95% of the variance in the data, thereby enhancing computational efficiency and mitigating the expensiveness of high dimensionality. The confusion matrix furthermore solidifies KNN as the most effective model tested in this project, as it has by-far the most weight on the diagonal, representing true predictions. The simplicity and flexibility of the KNN algorithm, coupled with appropriate model-preprocessing techniques, enabled us to achieve high levels of accuracy and reliability in predicting music genres of our sample snippets.

<u>SVM</u>



SVM achieved an accuracy of 0.7718 on the test dataset, demonstrating competitive performance among the models tested—although not quite beating the optimized KNN model. The weighted average of the precision, recall, and F1-score were each 0.77, indicating consistency and reliability in classifying music snippets across various genres. SVM exhibited an ability in handling high-dimensional data and multiclass classification, making it a generally suitable choice for our genre classification task. One particularly notable deliverable of SVM's performance is its confusion matrix. Upon inspection, the matrix reveals a strong weight-focus on the diagonal; the majority of predictions align with the true genre labels. This indicates that SVM effectively discerned patterns and characteristics specific to each genre, resulting in accurate classifications for most instances. Comparing SVM's confusion matrix to that of XGBoost, while both models exhibit a heavy diagonal presence, SVM stands out for its balanced distribution of accurate predictions across all genres. Unlike XGBoost, where certain genres, such as "blues", experienced a higher frequency of inaccurate predictions, SVM's predictions are more evenly distributed among the different genre labels, with no particular genre standing out as prone to misclassification. Overall, SVM's performance highlights its suitability for multiclass classification tasks, particularly in scenarios where data is high-dimensional and class boundaries are complex. Its ability to maintain a balanced distribution of accurate predictions across all classes further solidifies its position as a robust and reliable classifier for music genre classification tasks.

General

While each individual model showed promising results, our attempt to create an ensemble of all models performed poorly, achieving an accuracy of only just above 10%. This indicates that simply combining the predictions of multiple models may not necessarily lead to improved performance. This highlights the importance of careful algorithm selection and proper implementation of ensemble techniques.

In summary, KNN emerged as the top-performing model in terms of accuracy and overall predictive capability, followed closely by XGBoost and SVM. AdaBoost, while showing potential in ensemble learning, struggled to match the performance of other models in this particular context. These results highlight the importance of selecting appropriate machine learning algorithms and preprocessing techniques suited to the characteristics of the dataset. While some models may excel in certain scenarios, their effectiveness can vary significantly depending on the nature of the data and the task at hand.

Future Work

In the future, we can utilize other types of data to compare the predictions of our models. Right now, we examined a CSV file that was 30-second snippets split into 3 seconds to increase the rows of data. To validate and refine our models, we can use the full 30-second snippets. This will allow us to make observations of how larger datasets impact the accuracy and reliability of genre classification, as we will be able to see how our models perform in a more constrained data environment.

Additionally, we could leverage image data, specifically Mel Spectrograms. Mel Spectrograms are visualizations plotting the Mel Scale (measures human perception of pitch) vs. Time, demonstrating decibel levels through color intensity. By applying neural network techniques to these images, we will explore a different side of data analysis that we haven't worked with before. This use of image data can uncover hidden patterns that couldn't be easily identified using audio data. Using different types of data types, both audio and visual, we can develop more accurate genre classification models.

Conclusion

Our project allowed us to explore several different machine-learning algorithms by classifying music genres. Through KNN, SVM, and boosting methods like XGBoost and AdaBoost, we demonstrated that machine learning can effectively categorize music snippets into discrete genres with considerable accuracy. Supported by preprocessing techniques, our KNN model, optimized with SMOTE and PCA, was the most successful model we implemented. Our findings contribute to music streaming services' ability to better classify songs into genres, which can be used to improve user experience, like making personalized music recommendations. Also, the varying performances of our different models further highlight the importance of selecting appropriate models, implementing thorough preprocessing steps, and choosing optimized parameters.

Team Member Contribution

Over the course of this project, our team made sure to diligently distribute the workload equitably between individuals. To begin, we all met to brainstorm an idea for the project. After brewing some ideas, we set out on our own to find a few possible datasets to work with before meeting again to decide on the GTZAN dataset. Jake began the coding, completing the data preprocessing, exploratory data analysis, and the first few boost-based models. Veronica completed the more involved implementation of the KNN model. Finally, Devyanshi implemented the SVM model and our own ensemble model implementation. Meeting again, we all debugged, reorganized, and commented the code through as a team. GitHub was an essential resource for us during this time.

Upon completing the coding portion of the project, it was time to write the project report. Devyanshi began the report, completing the “Problem Description”, “Dataset”, and some of the “Approach and Methodology” portions. Jake then filled in the “Exploratory Data Analysis” section, also discussing the methods used to preprocess the data accordingly. Jake and Veronica worked together to complete the rest of the model subsections in the “Approach and Methodology” section. Jake worked together to complete the “Discussion and Result Interpretation” section, and then looked to Veronica to read over and edit the passages. Lastly, Devyanshi wrote the section regarding “Future Work” and Veronica wrote the “Conclusion”.

References

Beast, The Data. “AdaBoost, Gradient Boosting, XG Boost:: Similarities & Differences.” Medium, Medium, 7 Apr. 2023, https://medium.com/@thedatabeast/adaboost-gradient-boosting-xg-boost-similarities-differences-516874d644c6

“What Is XGBoost?” NVIDIA Data Science Glossary, 2024, https://www.nvidia.com/en-us/glossary/xgboost/.

Doshi, Ketan. “Audio Deep Learning Made Simple - Why Mel Spectrograms perform better.” Ketan Doshi Blog, 19 Feb. 2021, https://ketanhdoshi.github.io/Audio-Mel/