Music Genre Classification and Prediction

William Ryan wrryan@wisc.edu
Ryan Bali rbali@wisc.edu
Danny Miller djmiller23@wisc.edu

Abstract

Streaming platforms for music have increased tremendously in popularity over the last couple of years, and along with this comes the emergence of many new subsets of music genres, with people always trying to listen to the next new sound. In this paper, we aimed to fit different machine learning models to see which ones were most accurate in predicting the general musical genre of a song. We used a data set from Kaggle containing music from 10 different genres with 100 audio files each, all having a length of 30 seconds. We also tested for feature importance and analyzed which features of the audio files were most helpful in training different models to correctly predict music genres. We fit a variety of models to our data and performed hyperparameter tuning to yield high classification accuracy on our models. We experiment with Logistic Regression, Naive Bayes, Support Vector Machines, Random Forest, K-Nearest Neighbors, Gradient Boosting, Decision Trees, Histogram-based Gradient Boosting, and Bagging. We eventually ended up with four well performing models with a test accuracy of $> 87\%$*. Our best performing model (KNN) achieved a test accuracy of 92.8%.*

1. Introduction

As music has shifted towards digital platforms, the barriers to enter the industry as an artist have all but disappeared. Nearly any person with internet access can set up a Spotify account and start recording whatever song pops into their heads. In fact, as of February 2021, Spotify released reports stating that approximately 60,000 unique songs are uploaded onto their platform every day. Additionally, both music platforms and their artists benefit when a song becomes popular as both parties see increased revenue. Given that a song is much more likely to reach peak popularity if the track finds its optimal audience, it becomes clear how precise genre classification is an essential part of the music industry. Through the use of a quick and accurate classification model a company would be able to more easily sort through and distribute their ever growing library to the correct audience, thus leading to increased exposure for both their brand and their users.

In this study we explore the different paths of developing a supervised machine learning model that will be able to properly identify a song's genre. We have identified 1000 different 30 second audio clips that can be divided into 10 distinct musical categories. Using this data we hope to utilize machine learning methods such as k-nearest neighbors and random forests to identify common trends in the audio features for each style of music. This in turn will allow us to separate and sort future test data songs into their correct genre.

A similar study conducted by Roberto Basili, Alfredo Serafini, Armando Stellato at the Unirversity of Rome Tor Vergata [1] found that certain aspects of music did influence which genre a song would be filtered into. Some of these attributes included speed, volume and rhythm. We will make sure to take into account similar variables as we conduct our own research, as well as explore which technical aspects of these audio files are the most pivotal in associating a sound with a musical genre. Our goals for this project are to understand and communicate the underlying aspects of music that cause it to be classified into a specific genre. We will accomplish this through feature importance techniques and visualizations. We analyze the effectiveness of many multinomial classification models and rank which models perform the best. Specifically, we explore the use of models such as support vector machines (SVMs), K-Nearest Neighbor (KNN), Logistic Regression (LR), and Gradient Boosting (XGB). The end goal is to create a model that is able to reliably classify music according to their respective genre. We will determine the success of our model according to evaluation metrics such as accuracy, F-score, and AUC.

2. Related Work

The first piece of related work comes from Mohammad Tabrez Quasim, Eman H. Alkhammash, Mohammad Ayoub Khan & Myriam Hadjouni's [3] study on how to use machine learning and classification methods to make emotion-based music recommendations to music-streaming platform users. These researchers explored using IoT framework to build a model that would identify emotional information in a song. Training their model to identify certain sound, beats,

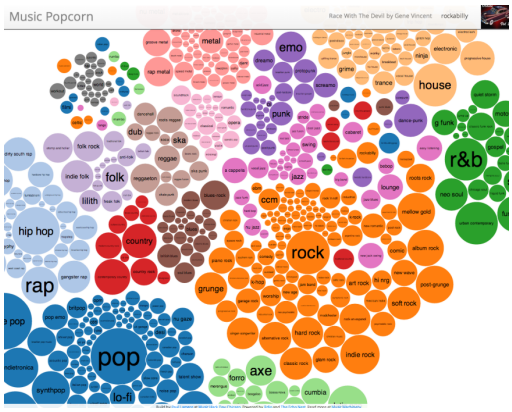

Figure 1. Example visualizing different music genres and how they connect via Music Popcorn. (Image source: https://musicmachinery.com/page/11/)

and instruments that are associated with different moods, the model would then be used to give users music recommendations directly on their streaming app like Spotify or Apple Music. Using machine learning methods like decision trees, deep cognitive systems, and K-nearest neighbor, they created an emotion-based music recommendation and classification framework that has a 96.12 percent prediction accuracy and precision rate of 96.69 percent. These levels have not been reached by existing approaches, and we plan to use decision trees, K-nearest neighbor, and other such machine learning models to try and build a classification model paired with hyperparameter tuning to achieve heightened accuracy.

Another related work comes from Professor Sebastian Raschka [4] and his study on creating a machine learning model to classify music by mood using song lyrics. Using Naive Bayes classification along with the GridSearch implementation in scikit-learn, a model was trained to classify songs based on their lyrics into two emotional categories - happy and sad. Precision and recall optimisation using the F1-score performance metric were the main focuses in developing the classifier, rather than just focusing on prediction accuracy, to remove sad songs from a playlist. The end goal was to pivot this into the development of a web app that could filter out sad songs from a users playlist if desired in order to influence their tendency to be in a better mood. The inspiration in this project to filter through different musical genres by identifying the most prominent features in the audio of the song was drawn from the work in mood identification in music by Professor Raschka. GridSearch and a confusion matrix to calculate F-scores are methods that will also be explored in our analysis in order to improve the accuracy in training our model.

3. Proposed Method

In this project, our group proposed to build different machine learning models in order to make predictions on a classifying songs into different genres, given a data set provided by Andrada Olteanu on Kaggle [2]. It contains 100 different 30 second samples of songs from 10 different genres in the Waveform Audio File format. Additionally, there is a dataset that provides 60 unique features such as tempo, Chroma STFT, and spectral centroid mean. The audio clips are 30 seconds each but we plan on decomposing them according to time segments obtained from the beginning, middle and end parts of the original music signal (time-decomposition). [5]

3.1. Resampling and Standardization

When building our models, a critical component was resampling and standardizing our data. We decided to use the 2-way holdout method for computing a test accuracy as well as repeated 5-fold cross validation. These would give us two ways of approximating our generalization accuracy. We aimed to use the 5-fold cross validation in addition to the 2-way holdout to find out if our 2-way holdout was providing a significant pessimistic bias.

When computing our 2-way holdout, we split our dataset two stratified samples. Our testing sample had 70% of the data and the testing sample had 30% of the data.

We additionally used standardization where it applied. Specifically, we used standardization on our KNN final model as well as our support vector machine model.

We used the sci-kit learn method StandardScaler to achieve a mean of zero on our dataset and a unit varience.

3.2. K-Nearest Neighbors

K-Nearest Neighbors or KNN is one of the most basic yet essential models in supervised machine learning. The model is considered simple due to it really only having one step in its training process which only involves storing the data that is processed through the classifier until it is necessary for a prediction to be made. Then, once the model has been fit, KNN will find the k nearest neighbors for each test data point and compute that data point's class label based on which label makes up the plurality of those k nearest neighbors.

For this model we will attempt to use both scaled and unscaled data, as well as testing both the Euclidean and Manhattan distance methods, Uniform and Distance weight variables, and several k values in order to obtain an optimal model.

3.3. Random Forest

Next we tested a random forest model. Random forests are very common in the machine learning world due to their

ease of use and relatively high results. Random forests are best described as bagging with decision trees. A number of decision trees will be fit onto bootstrap samples where a random subset of features will decide the optimal split at each node. This process will repeat according to the number of estimators until an optimal accuracy is achieved.

For random forest we tuned the model along its n_estimators, max_depth, criterion, oob_score, and warm_start parameters in order to achieve an optimal accuracy score. We felt that these parameters provided better results when changed from their original default setting. Additionally, we found the feature importance values for each of our variables in order to best determine which columns were having the largest effect when determining the class labels for our test data.

3.4. Gradient Boosting

Similar to random forests, gradient boosting is also a commonly used machine learning method. This model works by utilizing an ensemble of decision trees where new trees are added to fix the mistakes of the first generated trees. This process of fixing errors through newly generated decision trees continues until no further improvements can be made to the model. In a sense, gradient boosting combines numerous weak models to make a strong model.

With our gradient booster tuning we decided to analyze the n_estimators, learning_rate, alpha, lambda parameters as we figured they would have the largest effect on the outcome of our test data's accuracy.

3.5. Support Vector Machine

Finally we took a look at how the Support Vector Machine or SVM performed on our dataset. SVM is another supervised machine learning algorithm which attempts to define a hyperplane in an n-dimensional space, n being the number of features, that will separate each class label into its own defined space. The algorithm will continually insert hyperplanes into the space until a maximum distance between class label space's is found. The test data is then run against the model with this maximum distance.

We examined multiple types of SVM with our base model analysis including Linear SVM, Nu SVM and the normal SVM. Additionally, we tried out different decision_function_shape and max_iter.

4. Experiments

4.1. Visualization

Prior to building any models, we first wanted to understand the underlying data and if it was possible to model in an accurate manner. The first step was to simply visualize audio data. The first way we attempted to visualize data is using a mel spectogram as seen in Figure 2.

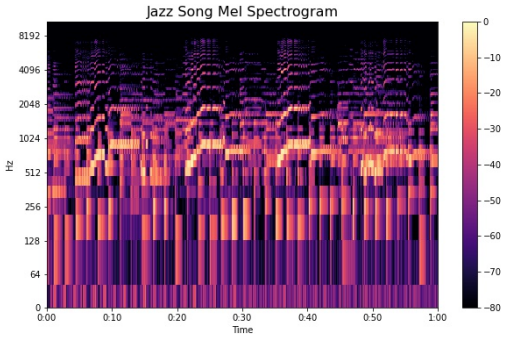

Figure 2. Mel Spectogram for a jazz song

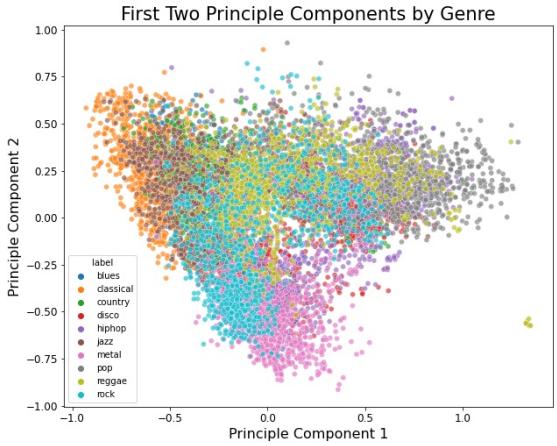

Figure 3. First two principle components by genre

However, now we wanted to see if it was possible to visualize the groupings of songs by genre. To do so we performed PCA to find the first two principle components. We found that the first two principle components explain 44.98% of the variance. By plotting the first two principle components we could clearly see there was a pattern in the data such that certain genres cluster together (Figure 3).

4.2. Models

The Logistic Regression, Naive Bayes, Support Vector Machines, Random Forest, K-Nearest Neighbors, Gradient Boosting, Decision Trees, Histogram-based Gradient Boosting, and Bagging machine learning models were all tested in a controlled manner to determine which model was most accurate in predicting the genre of a song. With the utilisation of Python libraries like scikit-learn and mlxtend, which are discussed in more detail later, we trained each model using their respective hyper-parameter settings in order to achieve higher prediction accuracies.

For this specific project our group was particularly interested in K-Nearest Neighbor and Random forest machine learning models, as they both have high capabilities in producing accurate models for both regression and classification problems, and don't need as much tuning as other approaches to produce very accurate results. Other models mentioned earlier such as gradient boosting in conjunction with tuning also have great tendency to produce accurate results and thus will be of equal importance while testing to obtain the best model.

4.3. Dataset

The Kaggle dataset [2] obtained for the musical genre analysis will consist of the following data fields and files:

- Genres Original: A collection of 10 genres with 100 audio files each, all having a length of 30 seconds
- Images Original: A visual representation for each audio file. One way to classify data is through Convolutional Neural Networks (CNN) because they usually take in some sort of image representation. If this method is chosen, the audio files are converted to Mel Spectrograms to make this possible
- 2 CSV files: Containing features of the audio files. The first file has a mean and variance computed over multiple features that can be extracted from an audio file for each song (30 seconds long). The second file has the same structure, but the each song was split into 3-second audio files, thus increasing the amount of data we input into our classification models tenfold

4.4. Software

Python, in conjunction with Jupyter Lab, will be the primary language used for modeling, feature engineering, and visualization. We will be utilizing the pandas and scikit-learn for our data handling and modeling procedures. We will also be using matplotlib and seaborn for data visualizations. Since we are dealing primarily with data in the Waveform Audio File format, we will be taking advantage of Python's librosa library to handle the music and audio analysis. Combined with matplotlib this will be particularly helpful for performing visualizations of audio data. Github was used to store all data sets and Jupyter notebooks for accessible viewing.

4.5. Hardware

Models were trained on MacBook Air and Dell personal computers.

5. Results and Discussion

The three evaluation metrics we used were accuracy, f1-score, and area under the receiver operating characteristic curve (ROC AUC).

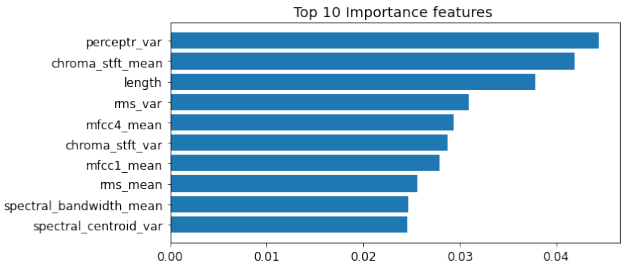

Figure 4. Visualization of our top 10 most important features from our feature importance algorithm

- **Accuracy**: The accuracy simply refers to the percentage of correctly identified test samples. We will use the test/validation accuracy to get a basic idea of our models performance.
- **F-score**: We can use our confusion matrix to calculate the precision and recall and find the harmonic mean to compute the F-score evaluation metric.

$$F_1 = 2\frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}} = \frac{\text{TP}}{\text{TP} + \frac{1}{2}(\text{FP} + \text{FN})}$$

- **AUC** In order to determine if our models can accurately distinguish between classes we will use AUC. We expect to achieve a AUC of higher than 0.5.

In order to get an approximation of our generalization accuracy for our models we use the 2-way holdout method and repeated k-fold cross validation.

5.1. Feature Importance

We thought a crucial part of our project is not only to gain an accurate model, but to understand out data as well. Accordingly, we performed feature importance on our dataset using two models that performed very well on our data. The first feature importance algorithm we ran is Random Forest Classification Feature Importance. The second is permutation feature importance using our K-Nearest Neighbors model. Fortunately, we found similar results in both cases.

We noticed that the top three most important features in all cases was the variance of the percussive components, the mean Chroma Frequencies, and the length. An interesting conclusion is that the percussive components contained much more predictive power as apposed the the harmonic components.

To explore the variance of the percussive components of the audio wave files in more detail we created a boxplot for

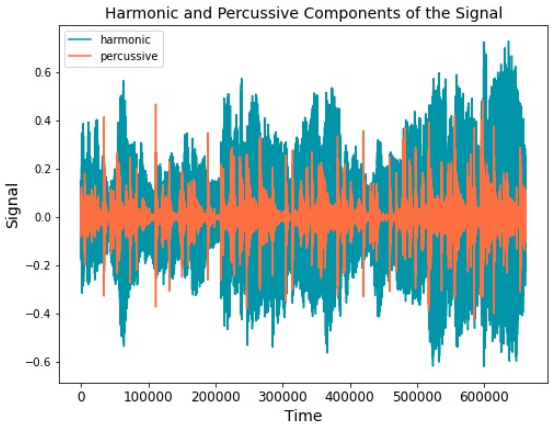

Figure 5. Visualization of the harmonic and percussive components of a country song

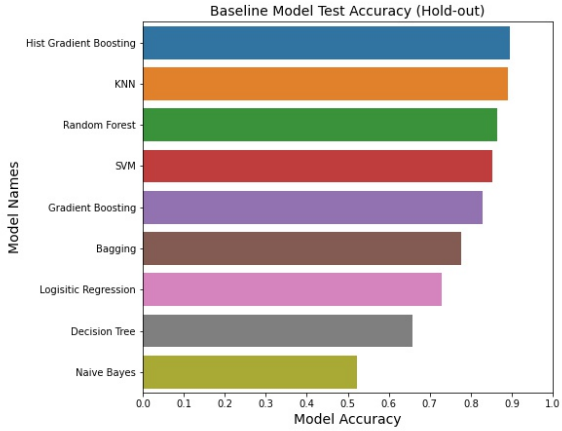

Figure 7. Test accuracy of the baseline models

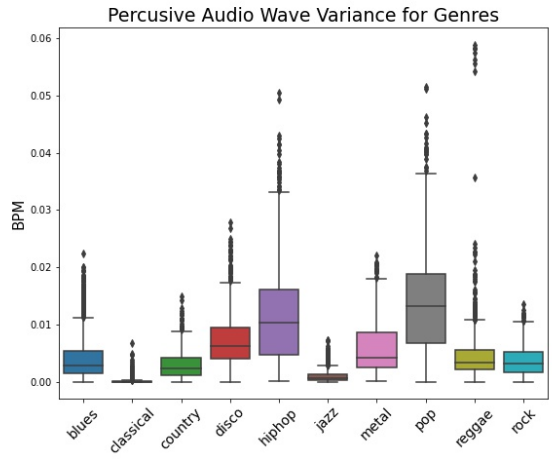

Figure 6. Boxplot of the Percussive components of the audio files for each genre

each of our ten genres. As we can see from Figure 6, there is a significant difference between each genre implying that it will provide significant predictive power especially for tree-based models.

5.2. Baseline Models

In order to identify which models perform the best on our data, we first used the 2-way holdout method to compute the test accuracy on our nine initial models. Our original models were K-Nearest Neighbors, gradient boosting, histogram based gradient boosting, random forest, Support Vector Machines, Bagging, Logistic Regression, Decision Tree, and Naive Bayes. By using all of these models we were able to get a general idea of the performance of many models so we could figure out which models to perform time-intensive hyper-parameter tuning on.

The results are shown in Figure 7 and as we can see the four best models are Histogram based gradient boosting, K-Nearest Neighbors, Random Forest, and Support Vector Machines. Now, using the test accuracy as heuristic of the generalization model performance we can perform hyper-parameter tuning on the following models to find the best performing model.

It is important to note that these baseline models will likely suffer from a pessimistic bias due to the fact that we performed a train-test split using 70% of the data for training and the remaining 30% of the data for testing. Therefore, in order to combat this, in the following sections we will use K-fold cross validation to reduce the impact of this pessimistic bias.

5.3. Support Vector Machine

To get the optimal Support Vector Machine model we used sklearn's GridSearchCV method. This performed Repeated 5-fold cross validation to tune the model parameters. The model parameters we tuned were the regularization parameter (C), gamma, and the kernel type. We were able to find the best cross validation accuracy using C = 100, gamma = 0.01, and the kernel is rbf. The corresponding cross validation accuracy was 88.71%. We also achieved an accuracy of 90.72% on the independent test set, an f1-score of 0.87214 and an AUC of 0.99429.

5.4. Random Forest

For our random forest model we also used a grid search based hyperparameter tuning method with 5-fold cross validation. We computed hyperparameter tuning on the number of trees in the forest, the max depth, the split function, and whether to use out-of-bag samples. The resulting parame-

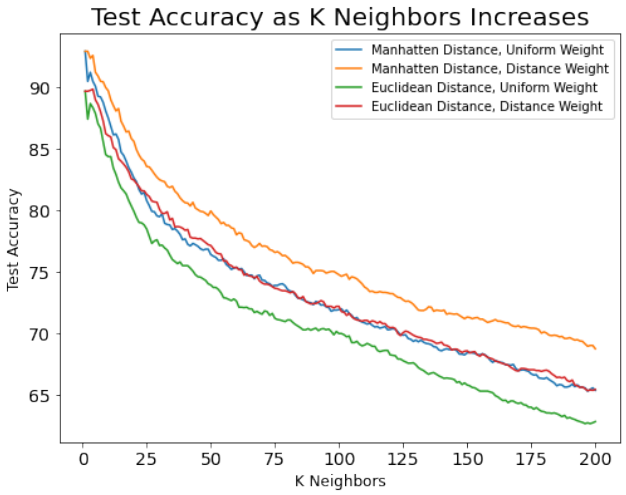

Figure 8. KNN model as we increase the value of k

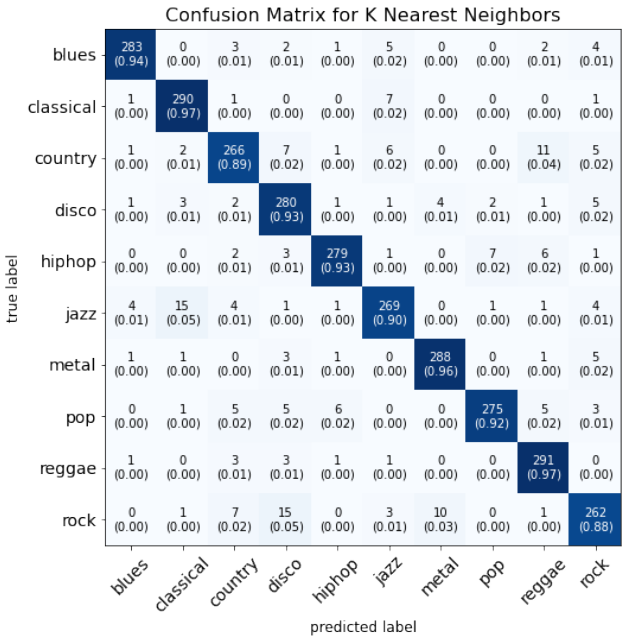

Figure 9. KNN model confusion matrix

ters we ended up with are the number of trees is 380, the max depth is 20, the split criterion is gini, and to use out-of-bag samples. Using these parameters we had a 5-fold cross validation score of 85.1567% and a test set accuracy of 87.2873%.

5.5. K-Nearest Neighbors

For our KNN model, we did grid search cross validation for hyperparameter tuning as well. We computed grid search on the number of neighbors, the weights, the algorithm, and the power parameter. We ended up with k being 1, the weights being uniform, the algorithm being kd-tree, and the power parameter is 1.

This model ended up performing quite well and being our model with the best test accuracy. We achieved a 5-fold cross validation score of 91.420% and a test accuracy of 92.759%. Although, we expected a lower k to overfit the model, it actually ended up performing very well on our test data. As we can see in Figure 8, as we increase the value k the accuracy consistently decreases.

We did note that although KNN has the highest in both test accuracy and F1-score, it has an AOC of 0.95976, which was the lowest of all of our four top performing models. As we know, the AUC is the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one. Therefore, we had to take a look at the confusion matrix for this model in Figure 9. However, after looking at the confusion matrix we did not notice any patterns that were a significant cause for concern.

5.6. Histogram-based Gradient Boosting

For our Histogram-based Gradient Boosting model we used the Sci-kit learn HistGradientBoostingClassifier. We also performed grid search repeated 5-fold cross validation for hyperparameter tuning and performance estimation. We tuned the model parameters learning rate and max number of leaf nodes. We found the best perfomance using a learning rate of 0.1 and the maximum leaf nodes set to 30.

This model achieved a 5-fold cross validation score of 89.032% and a test accuracy of 89.857%. This model also had a very high AUC of 0.99357.

5.7. Model Performance Comparison

As we can see below, all four of our models obtain a test accuracy of 87% and all models had an AUC of over 95%. Therefore, all four of our models likely can achieve a good generalization accuracy. Additionally, since we used the 2-way holdout method, we are likely encountering a slight pessimistic bias. However, due to the fact that the 5-way cross validation all received very similar accuracies to the test accuracy, we can assume that our generalization permanence is relatively accurate.

Based on the model performance chart is appears that KNN is our best model with a test accuracy of 92.76%. However, SVM also performed quite well with an accuracy of 90.72% as well as an AUC of 0.99429. Therefore, we assume either the KNN model or the SVM model both achieve good generalization accuracy on the dataset.

Model Performance			
	Test Accuracy	F1-Score	AUC
SVM	0.90724	0.90671	0.99429
Random Forest	0.87287	0.87214	0.98821
Gradient Boosting	0.89857	0.89834	0.99357
KNN	0.92759	0.92753	0.95976

When factoring in computation costs, KNN has a clear advantage in speed as it is a lazy learner. Additionally, random forest and gradient boosting are both quite time-intensive to train. However, it does have quite high memory requirements. Therefore, if memory is a limiting factor, SVM will be the recommended model. Otherwise KNN is the best option.

5.8. McNemar's Tests

In order to perform model selection we computed pairwise McNemar's tests with KNN, Gradient Boosting, and SVM. Since random forest has no real advantage over any of the other three models, we did not include it.

When comparing KNN and gradient boosting we got a p-value of $3.898 * 10^{-6}$ so we can reject the null hypothesis that the models are the same.

When comparing KNN and SVM we got a p-value of 0.00047 so we can also reject the null hypothesis that the models are the same.

However, when comparing gradient boosting and SVM we achieved a p-value of 0.1475. Therefore, we fail to reject the null hypothesis and we can assume the models are the same. Therefore, since gradient boosting is more computationally expensive on average, we see that the SVM model is a better choice between the two.

6. Conclusions

Through our analysis, we concluded that with relatively simple models such as KNN and SVM, we can achieve a generalization accuracy of $> 90\%$. However, there are many additional avenues for further research and experimentation in this area. One way, is by leveraging models such as deep neural networks to try and achieve better performance. Specifically, we could train a convolutional neural network on the Mel Spectrogram representation of the audio files to see if this technique may achieve better performance.

Another direction for more research would be to perform song similarity scoring. This has many applications such as improved recommendation engines. One way to do so would be using pairwise cosine similarity. However, there are a variety of different ways that this problem can be approached

7. Acknowledgements

We would like to recognize Andrada Olteanu who provided us with a very reliable data set [2] of audio files grouped by different genres on Kaggle.com. We would also like to thank Professor Raschka in helping us learn more about different machine learning models that would be most useful in creating a music classifier, and for all his teachings on the subject as we did not have a machine learning knowledge background prior to taking this class.

8. Contributions

As a group, it is decided that we are all taking responsibility to evenly divide computational and writing tasks. William was responsible for creating the baseline models and tuning them. Danny also worked on the models and created data visualizations. Ryan researched machine learning methods and helped create data visualizations. All three of us worked on creating, formatting, and proofreading the overall report prior to final submission. Computational tasks were conducted during scheduled meetings in College Library to code and group deadlines for different phases of the project were met to successfully generate a model that most accurately classifies and predicts the genre of different songs.

9. Code

The code and associated notebooks implemented for this project can be accessed at the following GitHub repository: `https://github.com/willryan1/Music-Genre-Classification`

References

[1] R. Basili, A. Serafini, and A. Stellato. Classification of musical genre: a machine learning approach. In *ISMIR*, 2004.

[2] A. Olteanu. Gtzan dataset - music genre classification. `https://www.kaggle.com/andradaolteanu/gtzan-dataset-music-genre-classification`, Mar 2020.

[3] M. T. Quasim, E. H. Alkhammash, M. A. Khan, and M. Hadjouni. Emotion-based music recommendation and classification using machine learning with iot framework. *Soft Computing*, pages 1–12, 2021.

[4] S. Raschka. Musicmood: Predicting the mood of music from song lyrics using machine learning. *arXiv preprint arXiv:1611.00138*, 2016.

[5] C. N. Silla, A. L. Koerich, and C. A. Kaestner. A machine learning approach to automatic music genre classification. *Journal of the Brazilian Computer Society*, 14(3):7–18, 2008.