What Lyrics Have to Say: Finding Correlation Between Lyrics and Genre

SENG 474 Fall 2019 Semester

December 4, 2019

Kristian Darlington · Becca Thomson

Table of Contents

List of Tables and Figures

1. Introduction

Throughout the last century and with the advent of mass media and globalization, the diversity of the music industry has increased substantially. Musical genres from hip-hop, which appeared in The Bronx during the 1970s [1], and country, which appeared in the southern US during the 1920s [2], have arisen to satisfy people's individualized cravings for different sounds. When asked what defines these genres, one might answer that it is the instruments or tools used, the tempo, or the environment in which a person listens to it.

This report explores the possibility that popular topics and terms suggested in song lyrics themselves are potential identifiers of a genre. Using an extensive dataset containing 380000 song lyrics, multiple machine learning algorithms will be trained and tested with the goal of determining the most identifiable genres based on language that is most correlated with a particular genre. This analysis is particularly interesting because the methods of text classification to be used do not take into account any of the "musical" aspects of each song; instrumental, temporal, and environmental factors are completely missing within the dataset.

Organized into well-known data mining tasks, this report details, in order, the preprocessing of data, the methods of evaluating models, and the mining of data. A conclusion finalizes the report, followed by references.

For the Python scripts used in this project, please email either Kristian Darlington at kristianfdarlington@gmail.com or Becca Thomson at beccabthomson@gmail.com.

2. Preprocessing

The dataset used in this project was sourced from the popular data mining website, Kaggle [3]. It contains the lyrics and related metadata for approximately 380000 songs spanning various genres; although this is a satisfactory amount of data for the purposes of this project, the set is not perfect and preprocessing was necessary to optimize it for the lyrical analysis.

2.1 Raw Data

Fields included within the `raw_lyrics.csv` file for each song include the name, year of release, artist, genre, and lyrics. To meet the requirements of this project, only the latter two fields, genre and lyrics, are necessary. Some duplicate songs exist in the dataset in which the lyrics are a combination of languages or a non-English language altogether.

2.2 Filtering and Manipulation

When reading the raw lyrics dataset into memory, the columns for name, year of release, and artist can immediately be dropped from the `pandas` dataframe to free up memory on the respective machine. These three columns are not necessary for training and testing machine learning models to predict song genre based solely on lyrics.

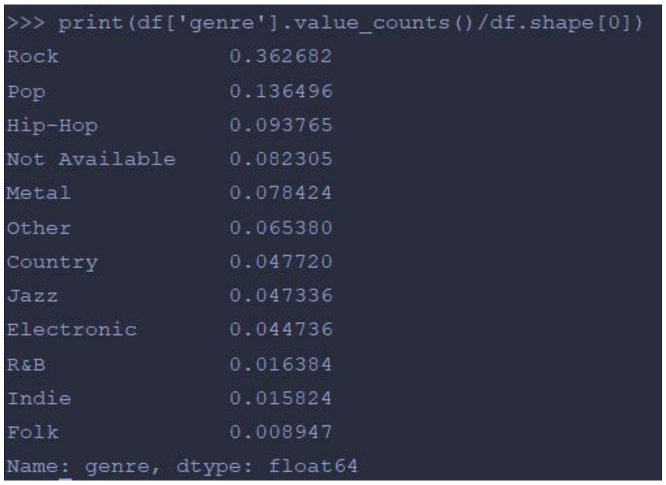
With only the genre and lyrics attributes remaining, an instance containing any missing field is not useful for developing the models in section 4. This means that rows where the genre or lyrics are not provided can be dropped from the dataset.

The first filter that is applied to each instance is the set of English words. The NLTK, or Natural Language Toolkit, Python library provides extensive support for manipulating English text, including a set of all English words and sentence parsing capabilities [4]. The models in this project focus on English songs for simplicity, so foreign words were removed if they did not exist in the set of English words. During this process, all punctuation and numerics were removed from the lyrics; these include all characters that do not exist in the English alphabet. This data preprocessing step ensures that the sole remaining lyrical data involves meaningful English words and nothing else.

Since the analysis is looking for words in song lyrics that are unique to a genre and common within that genre, stop words add noise to the dataset and provide little, if any, gain regarding the predictive accuracy of the models. Stop words include those with a generally high frequency of use in natural language, such as “a,” “the,” and “and.” To deal with this, the `TfidfVectorizer` class provided by Scikit-learn to vectorize lyric data was used to eliminate words that appeared in more than 80% of documents before training and testing the models. This would eliminate all stop words and more.

2.3 Balancing the Training Set

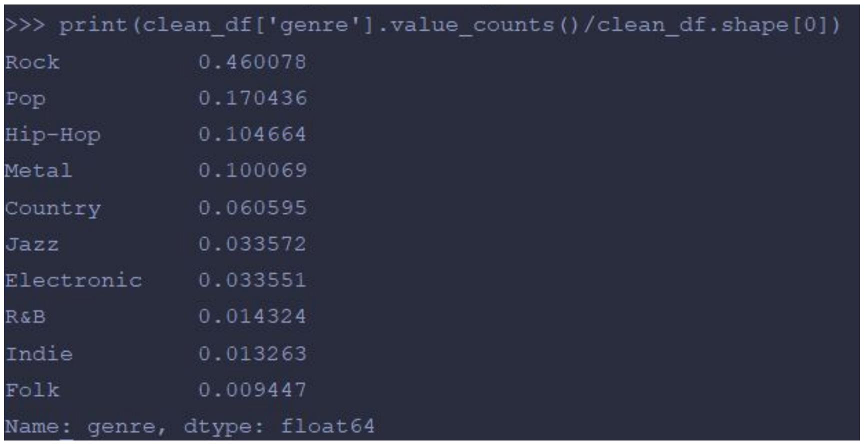
Of the original instances sourced from the Kaggle lyrics dataset, 36% are ‘Rock’ and 14% ‘Pop’, with the remaining 50% of instances divided amongst 10 other genres (see Figure 1).



```
>>> print(df['genre'].value_counts()/df.shape[0])
Rock             0.362682
Pop              0.136496
Hip-Hop          0.093765
Not Available    0.082305
Metal            0.078424
Other            0.065380
Country          0.047720
Jazz             0.047336
Electronic       0.044736
R&B              0.016384
Indie            0.015824
Folk             0.008947
Name: genre, dtype: float64
```

Figure 1: The percentage of genre occurrence in the dataset before preprocessing.

After processing the data as described above, the resulting dataset contains approximately 46% ‘Rock’ instances and 17% ‘Pop’ instances, with even lower measures for other genres. Exact values are shown below (see Figure 2).



```
>>> print(clean_df['genre'].value_counts()/clean_df.shape[0])
Rock          0.460078
Pop           0.170436
Hip-Hop       0.104664
Metal         0.100069
Country       0.060595
Jazz          0.033572
Electronic    0.033551
R&B           0.014324
Indie         0.013263
Folk          0.009447
Name: genre, dtype: float64
```

Figure 2: The percentage of genre occurrence in the dataset after preprocessing.

Given this unbalanced training set, the models used during data mining produce poor predictions due to favouring the majority class (‘Rock’). Since the majority class accounts for nearly half of the dataset, the models can minimize the overall error rate and achieve nearly 50% accuracy simply by predicting ‘Rock’ on any given input.

To solve this issue, the parameter `class_weight='balanced'` was used with the classifiers whenever the classifier offered that functionality (if the classifier did not, as in the case of Scikit-learn’s MultinomialNB, the data was balanced manually). The `class_weight` parameter describes a method of assigning each instance a weight that is the inverse of its frequency in the dataset, so that instances of rarer classes have a greater impact on the loss function and vice versa.

The `TfidfVectorizer` was used to balance the count vectors for each individual term, or attribute, by using a formula described as “term-frequency multiplied by inverse document-frequency” (see Figure 3) [5].

$$\text{tf-idf}(t,d) = \text{tf}(t,d)*\ln[(1 + n)/(1 + \text{df}(d,t))] + 1$$

Figure 3: The equation that Scikit-learn’s `TfidfVectorizer` applies to attribute counts, where *t* indicates term, *d* indicates document, and *n* is term-frequency.

This equation is important in the preprocessing of the dataset because not all song lyrics are of the same length; naturally, longer lyrics will contain higher counts for terms than shorter lyrics and this should be compensated for.

2.4 Splitting Data into Training and Testing Sets

The `train_test_split()` function provided in the Scikit-learn library is a convenient tool for splitting the preprocessed lyrical dataset into a training set and a testing set for use with the models. For model training and testing in this project, the dataset was split into 85% training instances and 15% testing instances at random.

3. Evaluation

The two key metrics used to evaluate each model detailed in section 4 are the model F1 scores and accuracies.

Accuracy is a measure of a model's overall success, measured by the ratio of correct predictions to total attempted predictions. When measuring accuracy, precision and recall are not considered, as they are with the F1 score. Although accuracy is a convenient way to gauge the performance of a classifier, it is important to note that it can be biased and misleading. As detailed in section 2.3, when the models were trained and tested on the lyrics dataset before the instances in that dataset were balanced by genre ('Rock' comprising 50% of the samples), a 50% accuracy was achieved by trivially predicting 'Rock' each time.

To obtain a better understanding of the model performance, the weighted- and macro-F1 scores were used. An F1 score for a single class is the harmonic mean of precision and recall for that class, computed by the following equations (see Figure 4).

$$\text{F1 score} = 2*p*r/(p + r)$$

$$p = \text{true positives} / (\text{true positives} + \text{false positives})$$

$$r = \text{true positives} / (\text{true positives} + \text{false negatives})$$

Figure 4: The equations defining F1 score, precision, and recall, respectively; *p* indicates precision and *r* indicates recall.

A single F1 score only provides a value for binary classifications. To modify this for higher order classification, an F1 score had to be calculated for each genre in the test set. For the macro-F1 score defining the performance of the models in a single value, the simple average of F1 scores for each genre was calculated. For the weighted-F1 scores defining the performance of the models in a single value, the weighted average of F1 scores for each class was calculated; the weights applied to each F1 score were determined by number of instances of each class in the test set.

Like accuracy, F1 scores should be used with care. The calculation gives the same weight to precision and recall, which often does not represent the nature of the problem domain. Despite this, it was deemed valid for lyric classification into genres because a false positive has the same implications as a false negative in the problem domain. If this report were to detail sensitive data like illness diagnosis, however, the equal weighting given to precision and recall would be inappropriate.

4. Data Mining

The data mining portion of this project pertains to the training and testing of various machine learning models; the goal of this process is to find a model that best predicts the genre of a given song based on the lyrics provided. The following sections are organized with respect to the model described in that section, each with their code borrowed from the Python Scikit-learn library.

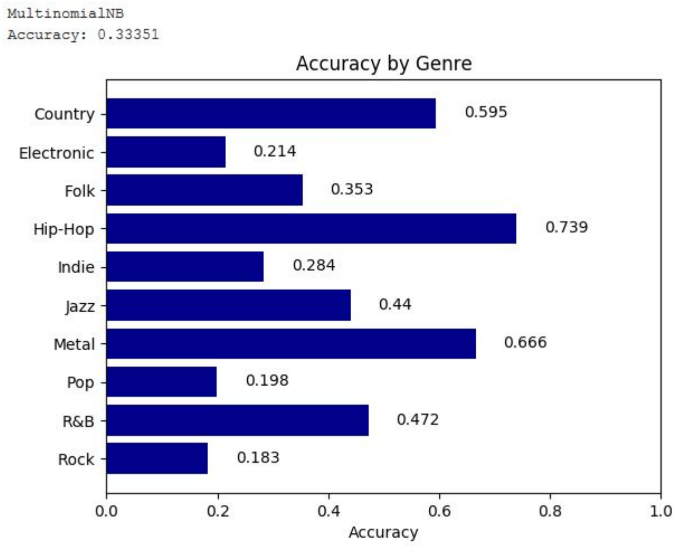
Sections 4.1, 4.2, 4.3, and 4.4 evaluate each classifier based on simple accuracy measures to obtain a general idea of the most identifiable genres of music in the dataset. Section 4.5, however, narrows the scope of the classification based on the information obtained and runs each model again, then provides the more insightful F1 score, precision, and recall using the results of this. For insightful results of each classifier on the full dataset, see section 4.6 where a preferred classifier is chosen.

4.1 Multinomial Naive Bayes

The Naive Bayes classifier is a common and simple machine learning model that classifies inputs based on probabilistic calculations involving the frequency-of-appearance of attributes in the training set provided. It assumes the highly unlikely case of complete independence between the attributes provided to it, but often performs satisfactorily despite this. Multinomial Naive Bayes is a variation of the Naive Bayes classifier that uses a multinomial distribution for calculating probabilities as opposed to a Gaussian distribution, which is geared towards the text analysis being analysed.

The `class_weight` parameter mentioned in section 2.3 does not exist with the Multinomial Naive Bayes classifier provided by Scikit-learn. As a result, the documents in the training set had to be manually duplicated where necessary; this task is performed to remove the future prediction bias given towards the classes ‘Rock’ and ‘Pop’ due to those classes making up approximately 60% of the training dataset, while the remaining classes account for the leftover 40%. The goal of manually balancing the training data is to have an approximately equal number of instances per class such that the model is not biased. The `RandomOverSampler` class from the imblearn package was used to achieve this balance [6].

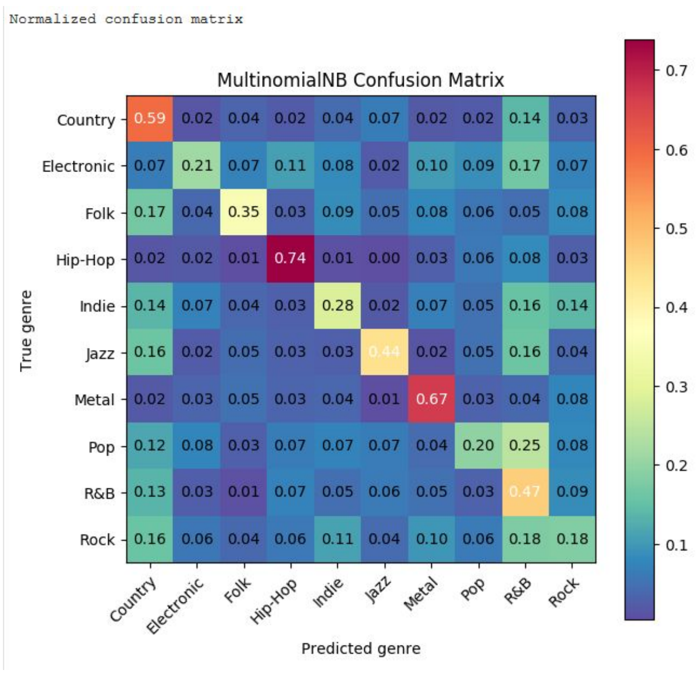
The overall accuracy of the Multinomial Naive Bayes classifier was shown to be 33.35%. The histogram corresponding to these results is shown below (see Figure 5).



```
MultinomialNB
Accuracy: 0.33351
```

Figure 5: A histogram depicting the Multinomial Naive Bayes classifier's prediction accuracy, by genre.

A confusion matrix elaborating on the above histogram can be found below (see Figure 6).



```
Normalized confusion matrix
```

True genre \ Predicted genre	Country	Electronic	Folk	Hip-Hop	Indie	Jazz	Metal	Pop	R&B	Rock
Country	0.59	0.02	0.04	0.02	0.04	0.07	0.02	0.02	0.14	0.03
Electronic	0.07	0.21	0.07	0.11	0.08	0.02	0.10	0.09	0.17	0.07
Folk	0.17	0.04	0.35	0.03	0.09	0.05	0.08	0.06	0.05	0.08
Hip-Hop	0.02	0.02	0.01	0.74	0.01	0.00	0.03	0.06	0.08	0.03
Indie	0.14	0.07	0.04	0.03	0.28	0.02	0.07	0.05	0.16	0.14
Jazz	0.16	0.02	0.05	0.03	0.03	0.44	0.02	0.05	0.16	0.04
Metal	0.02	0.03	0.05	0.03	0.04	0.01	0.67	0.03	0.04	0.08
Pop	0.12	0.08	0.03	0.07	0.07	0.07	0.04	0.20	0.25	0.08
R&B	0.13	0.03	0.01	0.07	0.05	0.06	0.05	0.03	0.47	0.09
Rock	0.16	0.06	0.04	0.06	0.11	0.04	0.10	0.06	0.18	0.18

Figure 6: A confusion matrix for the Multinomial Naive Bayes classifier.

By the histogram and confusion matrix, it can be seen that the most identifiable genres for the Multinomial Naive Bayes classifier are 'Hip-Hop', 'Metal', and 'Country', with the genre most mistaken for being 'Rock'.

4.2 Linear Support Vector Classification

A Support Vector Machine (SVM) is a supervised machine learning model commonly used for text classification. SVMs attempt to determine a hyperplane that best divides a dataset into two classes. Multiclass SVMs also use this binary approach, but they do so by dividing the overall dataset into multiple binary problems.

A Linear SVM uses a linear kernel to split the instances; the `LinearSVC` Scikit-learn classifier was used to classify the data using this method. By default, the model uses an `'ovr'` scheme, which trains "one-versus-rest" classifiers for each class, as well as a `'squared_hinge'` loss function. As the number of samples was greater than the number of features, `dual=False` was set, as recommended in the Scikit-learn documentation [7]. This selects the algorithm to solve the primal optimization problem, rather than the dual one. As discussed in section 2.3, the `class_weight='balanced'` parameter was set to account for unbalanced data.

The overall accuracy of the LinearSVC classifier was shown to be 47.41%. The histogram corresponding to these results is shown below (see Figure 7).

```
LinearSVC
Accuracy: 0.4741
```

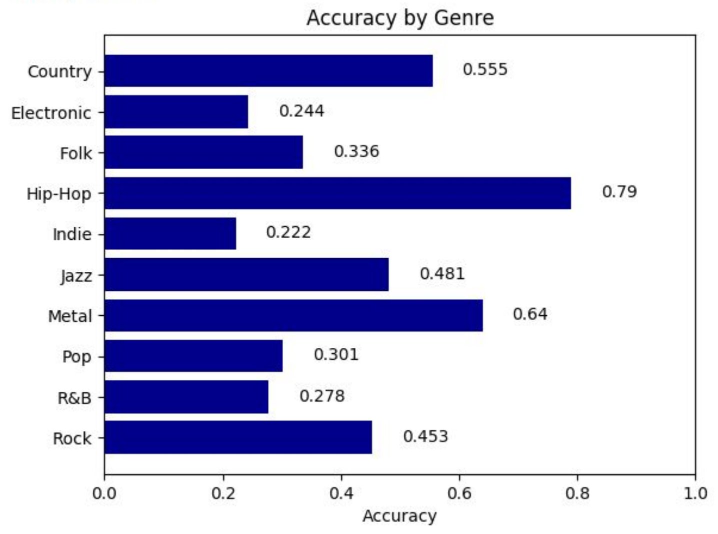
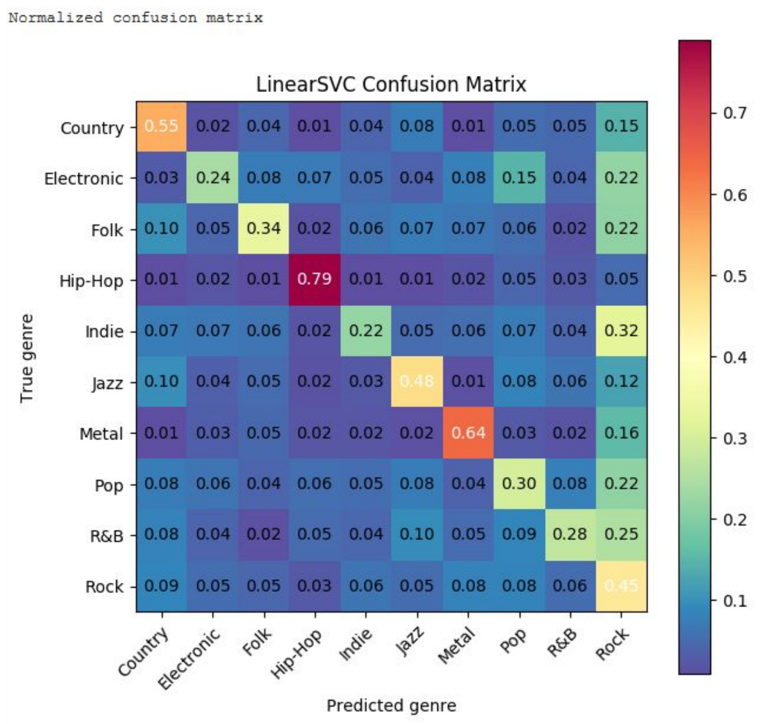


Figure 7: A histogram depicting the Linear SVC classifier's prediction accuracy, by genre.

A confusion matrix elaborating on the previous histogram can be found below (see Figure 8).



```
Normalized confusion matrix
```

True genre \ Predicted genre	Country	Electronic	Folk	Hip-Hop	Indie	Jazz	Metal	Pop	R&B	Rock
Country	0.55	0.02	0.04	0.01	0.04	0.08	0.01	0.05	0.05	0.15
Electronic	0.03	0.24	0.08	0.07	0.05	0.04	0.08	0.15	0.04	0.22
Folk	0.10	0.05	0.34	0.02	0.06	0.07	0.07	0.06	0.02	0.22
Hip-Hop	0.01	0.02	0.01	0.79	0.01	0.01	0.02	0.05	0.03	0.05
Indie	0.07	0.07	0.06	0.02	0.22	0.05	0.06	0.07	0.04	0.32
Jazz	0.10	0.04	0.05	0.02	0.03	0.48	0.01	0.08	0.06	0.12
Metal	0.01	0.03	0.05	0.02	0.02	0.02	0.64	0.03	0.02	0.16
Pop	0.08	0.06	0.04	0.06	0.05	0.08	0.04	0.30	0.08	0.22
R&B	0.08	0.04	0.02	0.05	0.04	0.10	0.05	0.09	0.28	0.25
Rock	0.09	0.05	0.05	0.03	0.06	0.05	0.08	0.08	0.06	0.45

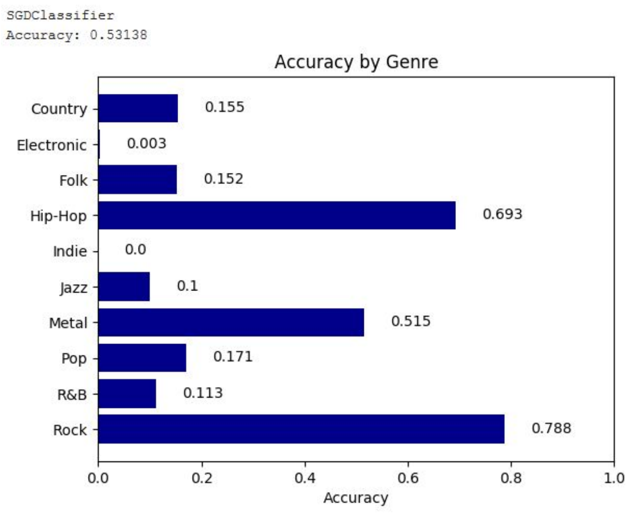
Figure 8: A confusion matrix for the Linear SVC classifier.

By the histogram and confusion matrix, it can be seen that the most identifiable genres for the Linear SVC classifier are 'Hip-Hop', 'Metal', 'Country', and 'Jazz', with the genre most mistaken for being 'Rock'. These results agree with the Multinomial Naive Bayes classifier, although the overall accuracy is significantly better.

4.3 Linear Support Vector Classification with Stochastic Gradient Descent

The Linear SVM, like Scikit-learn's `LinearSVC` class, can also be used with Stochastic Gradient Descent (SGD). This estimator samples the loss gradient after each training instance and updates the model accordingly. After performing a basic grid search using `GridSearchCV`, the parameters selected were `alpha=0.001` and `loss='log'`.

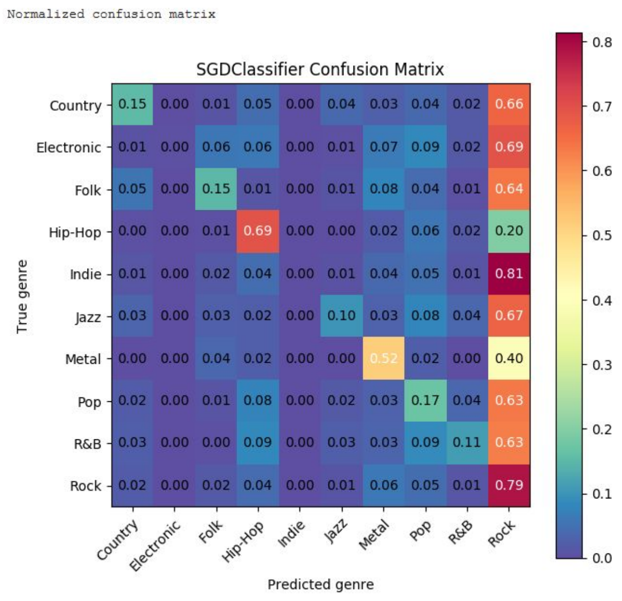
The overall accuracy of the Linear Support Vector Classification with Stochastic Gradient Descent was found to be 53.14%. The histogram below depicts the accuracy per genre for this model (see Figure 9).



```
SGDClassifier
Accuracy: 0.53138
```

Figure 9: A histogram depicting the Linear SVM with SGD classifier's prediction accuracy, by genre.

A confusion matrix elaborating on the above histogram can be found below (see Figure 10).



```
Normalized confusion matrix
```

	Country	Electronic	Folk	Hip-Hop	Indie	Jazz	Metal	Pop	R&B	Rock
Country	0.15	0.00	0.01	0.05	0.00	0.04	0.03	0.04	0.02	0.66
Electronic	0.01	0.00	0.06	0.06	0.00	0.01	0.07	0.09	0.02	0.69
Folk	0.05	0.00	0.15	0.01	0.00	0.01	0.08	0.04	0.01	0.64
Hip-Hop	0.00	0.00	0.01	0.69	0.00	0.00	0.02	0.06	0.02	0.20
Indie	0.01	0.00	0.02	0.04	0.00	0.01	0.04	0.05	0.01	0.81
Jazz	0.03	0.00	0.03	0.02	0.00	0.10	0.03	0.08	0.04	0.67
Metal	0.00	0.00	0.04	0.02	0.00	0.00	0.52	0.02	0.00	0.40
Pop	0.02	0.00	0.01	0.08	0.00	0.02	0.03	0.17	0.04	0.63
R&B	0.03	0.00	0.00	0.09	0.00	0.03	0.03	0.09	0.11	0.63
Rock	0.02	0.00	0.02	0.04	0.00	0.01	0.06	0.05	0.01	0.79

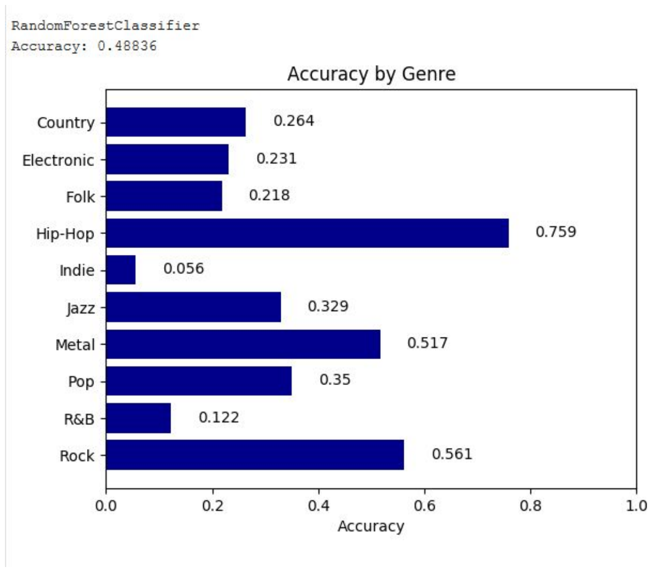
Figure 10: A confusion matrix for the Linear SVM with SGD classifier.

By the histogram and confusion matrix, it can be seen that the most identifiable genres for the Linear SVM classifier with SGD are 'Hip-Hop' and 'Rock', with the genre most mistaken for being 'Rock'. These results somewhat agree with the other classifiers used in this project.

4.4 Random Forest Classifier

The Random Forest Classifier creates a number of decision trees based on randomly selected data samples, then, when run against a test set, classifies each instance based on a majority vote between these trees. The Random Forest Classifier was used with parameters `max_depth=40, n_estimators=15`. While accuracy could have been increased by allowing a greater number of estimators or a greater depth, these parameters balanced the desire for accuracy with efficiency.

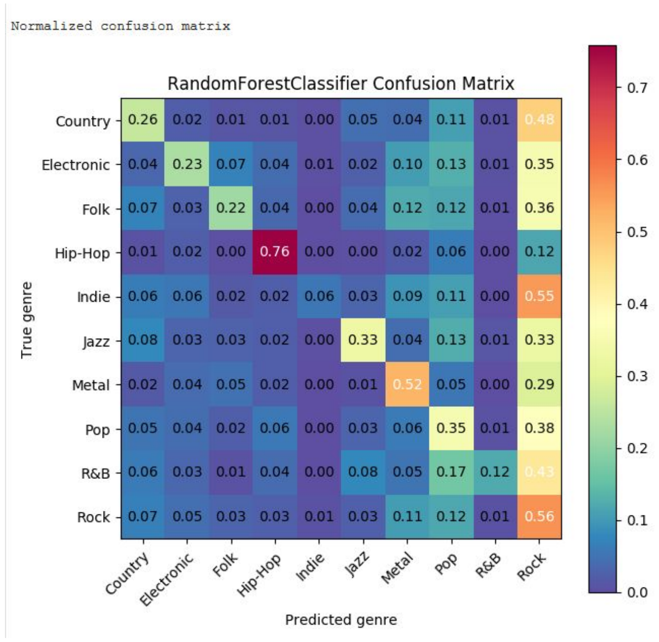
The overall accuracy of the Random Forest Classifier was found to be 48.38%. The histogram below summarizes the model's performance classifying each genre (see Figure 11).



```
RandomForestClassifier
Accuracy: 0.48836
```

Figure 11: A histogram depicting the Random Forest classifier's prediction accuracy, by genre.

A confusion matrix elaborating on the above histogram can be found below (see Figure 12).



```
Normalized confusion matrix
```

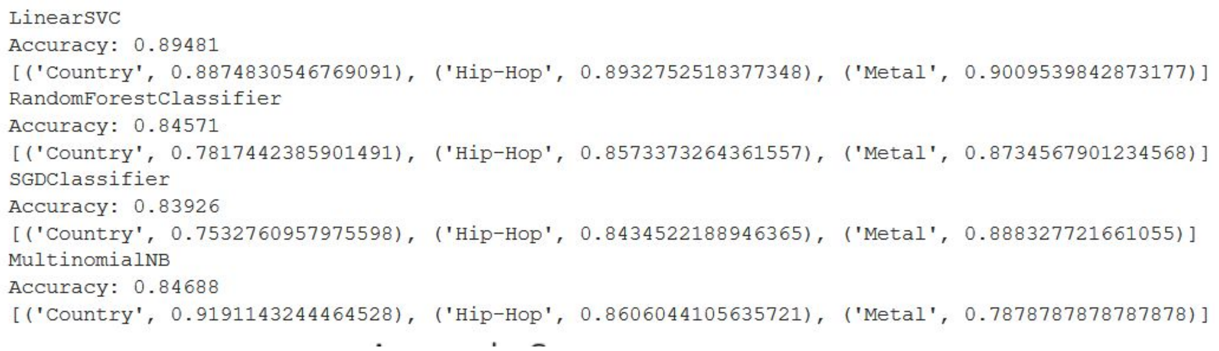
Figure 12: A confusion matrix for the Random Forest classifier.

By the confusion matrix for this classifier, the most recognizable genres were ‘Hip-Hop’, ‘Rock’, and ‘Metal’, with the genre most mistaken for remaining ‘Rock’. These results coincide with the other classifiers discussed above.

4.5 Observations

Across all classifiers used, three genres stood above the rest in their recognizability: ‘Hip-Hop’, ‘Country’, and ‘Metal’. A trend visible in the results was the genre most mistaken for being ‘Rock’. Using this information, the dataset can be trimmed further to explore each classifier’s ability to distinguish the most identifiable genres from one another, excluding the rest. This is done to evaluate the performance of the classifiers without the “noise” that genres like ‘Rock’ introduce.

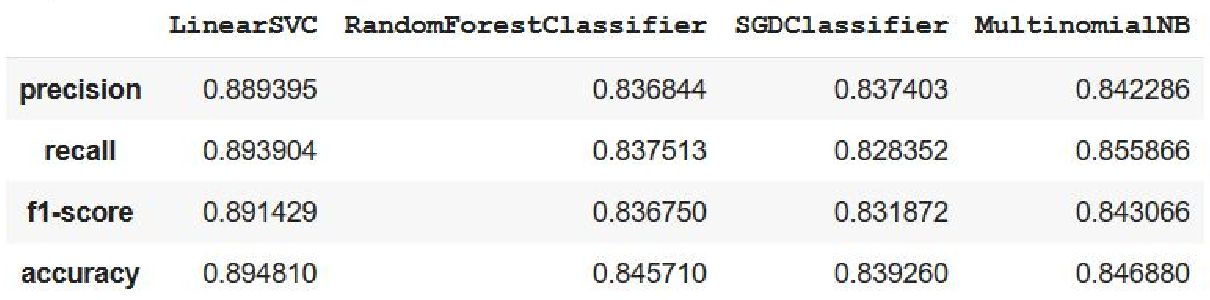
The models were executed a second time using instances from the three classes with the greatest recognizability and the results are seen below (see Figure 13).



```
LinearSVC
Accuracy: 0.89481
[('Country', 0.8874830546769091), ('Hip-Hop', 0.8932752518377348), ('Metal', 0.9009539842873177)]
RandomForestClassifier
Accuracy: 0.84571
[('Country', 0.7817442385901491), ('Hip-Hop', 0.8573373264361557), ('Metal', 0.8734567901234568)]
SGDClassifier
Accuracy: 0.83926
[('Country', 0.7532760957975598), ('Hip-Hop', 0.8434522188946365), ('Metal', 0.888327721661055)]
MultinomialNB
Accuracy: 0.84688
[('Country', 0.9191143244464528), ('Hip-Hop', 0.8606044105635721), ('Metal', 0.7878787878787878)]
```

Figure 13: Model accuracy by genre when limited to ‘Country’, ‘Hip-Hop’, and ‘Metal’.

The data shows that the Linear SVC model can achieve nearly 90% accuracy when limited to these three genres. Interestingly, ‘Metal’ is the best-classified genre for all but the Multinomial Naive Bayes model, which classifies ‘Country’ most successfully. Performance metrics that provide more insight regarding model performance, like the F1 score, are shown below for the trial, in which only three genres were classified (see Figure 14).



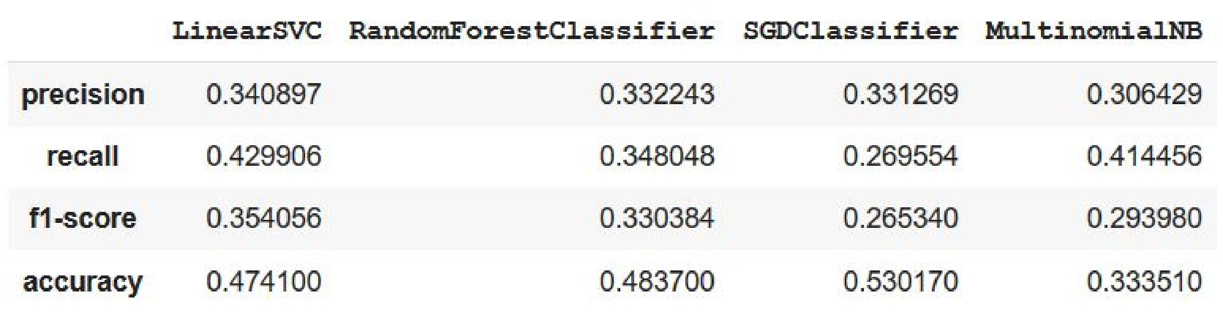
	LinearSVC	RandomForestClassifier	SGDClassifier	MultinomialNB
precision	0.889395	0.836844	0.837403	0.842286
recall	0.893904	0.837513	0.828352	0.855866
f1-score	0.891429	0.836750	0.831872	0.843066
accuracy	0.894810	0.845710	0.839260	0.846880

Figure 14: Classifier performance metrics when genres are limited to ‘Country’, ‘Hip-Hop’, and ‘Metal’.

As seen above, the model with the highest F1 score and accuracy when run on the dataset of only three genres was the Linear SVC.

4.6 Preferred Classifier

The F1 score, precision, recall, and accuracy metrics described in sections 4.1 through 4.4 are summarized in the following table for each classifier (see Figure 15).



	LinearSVC	RandomForestClassifier	SGDClassifier	MultinomialNB
precision	0.340897	0.332243	0.331269	0.306429
recall	0.429906	0.348048	0.269554	0.414456
f1-score	0.354056	0.330384	0.265340	0.293980
accuracy	0.474100	0.483700	0.530170	0.333510

Figure 15: Summary of precision, recall, F1 score, and accuracy for each classification model.

While the Support Vector Classification using Stochastic Gradient Descent evidently produces the greatest accuracy, the Linear SVC model achieves the highest F1 score, which makes up for the slightly lower accuracy. For this reason, the Linear SVC model is considered the preferred classifier. The results of the models when trained and tested on the three most recognizable genres supports this decision, as in section 4.5, since the accuracy and F1 score of the Linear SVC model are significantly higher than those of the other models.

5. Conclusion

Results gathered from this experiment show that the recognizability of different musical genres based on song lyrics alone is heavily dependent on the genres contained in the training and testing sets of the chosen classifier. As discussed in section 4.5, the F1 scores and accuracies for the models used were remarkably good when classifying lyrics into ‘Hip-Hop’, ‘Country’, and ‘Metal’, but slightly less remarkable when classifying lyrics into a set of genres that include ‘Rock’. These results suggest a lack of defining words or recurring topics in the rock genre when compared to other, apparently niche genres. In turn, this may suggest that themes present in rock are a conglomerate of each other major genre. This is supported by the historical connection between rock, metal, country, and jazz [8]. Many may consider metal, country, and jazz to be subsections of the encompassing genre of rock, yet unique genres amongst one another; this project would agree with that intuition.

For the purposes of this project, the classifier that produced the best results overall was the Linear Support Vector Classifier, or Scikit-learn’s `LinearSVC` class, based on performance presented in section 3. This would agree with the data mining communities’ general consensus regarding the question of what classifier is best for text classification. Popular classifiers for text analysis include Multinomial Naive Bayes and Support Vector Machines, but Multinomial Naive Bayes, while simpler and faster, is often outperformed by SVMs [9].

References

[1] J. Chang, *Can't Stop Won't Stop: A History of the Hip-Hop Generation*. S.l.: Picador, 2020.

[2] A. A. Fox and R. A. Peterson, "Creating Country Music: Fabricating Authenticity," *Notes*, vol. 55, no. 1, p. 127, 1998.

[3] GyanendraMishra, "380,000 lyrics from MetroLyrics," *Kaggle*, 11-Jan-2017. [Online]. Available: https://www.kaggle.com/gyani95/380000-lyrics-from-metrolyrics. [Accessed: 15-Nov-2019].

[4] "Natural Language Toolkit," *Natural Language Toolkit - NLTK 3.4.5 documentation*, 20-Aug-2019. [Online]. Available: https://www.nltk.org/index.html. [Accessed: 20-Nov-2019].

[5] "sklearn.feature_extraction.text.TfidfVectorizer," *scikit*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html. [Accessed: 26-Nov-2019].

[6] "imblearn.over_sampling.RandomOverSampler," *imbalanced*, 2016. [Online]. Available: https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.over_sampling.RandomOverSampler.html. [Accessed: 22-Nov-2019].

[7] "sklearn.svm.LinearSVC," *scikit*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html. [Accessed: 18-Nov-2019].

[8] W. E. Studwell and D. F. Lonergan, *The Classic Rock and Roll Reader: Rock Music From its Beginnings to the Mid-1970s*. London: Haworth, 2000.

[9] M. Guia, R. Rocha Silva, and J. Benardino, "INSTICC Portal," *INSTICC Portal*, 17-Sep-2019. [Online]. Available:

http://www.insticc.org/node/TechnicalProgram/ic3k/presentationDetails/83641.
[Accessed: 30-Nov-2019].