Project 5: Predicting Explicit Lyrics

Project 5 Presentation [GitHub Repository]

Introduction

For this project, we wanted to see if we could train a model to distinguish between an explicit and a non-explicit song given its lyrics. A good model can help music companies more efficiently determine that rating and more effectively protect their customers from accidentally listening to music they prefer not to hear. To build this model, we used data from a company called Musixmatch, which provides lyrics between several platforms, along with the Naive Bayes classifier module from the scikit-learn library. Above, we have included links to the GitHub repository which contains the source code for this project, as well as the presentation slides for the project.

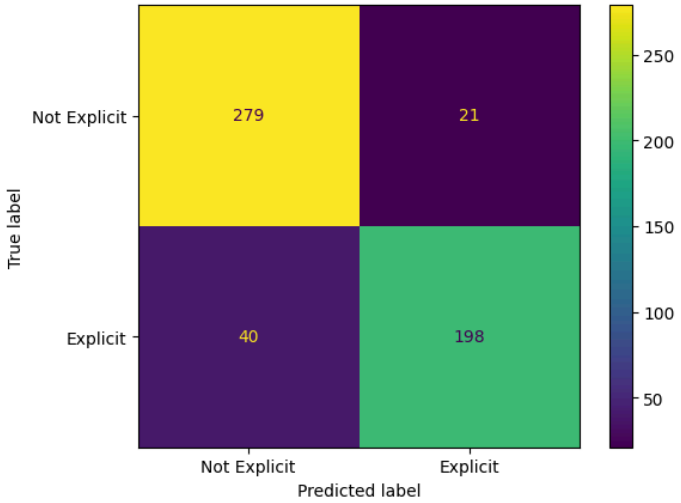
Dataset

For our dataset, we used Musixmatch's API to pull the lyrics of songs. We decided to focus on chart-topping artists in America, as these artists provide a good mixture of explicit and non-explicit songs, and most songs made by these artists feature English lyrics. We used an endpoint to get the top 13 artists in the United States, and then got all albums of each of those artists, and all songs of each album. The API call included each song's lyrics, as well as an explicit flag, which had a value of 0 if the song was not explicit and a value of 1 if the song was explicit. Since Musixmatch is directly used to display lyrics by Spotify and Apple Music, two of the biggest music-streaming platforms, we decided that that would likely be the most extensive and trustworthy dataset for our purposes.

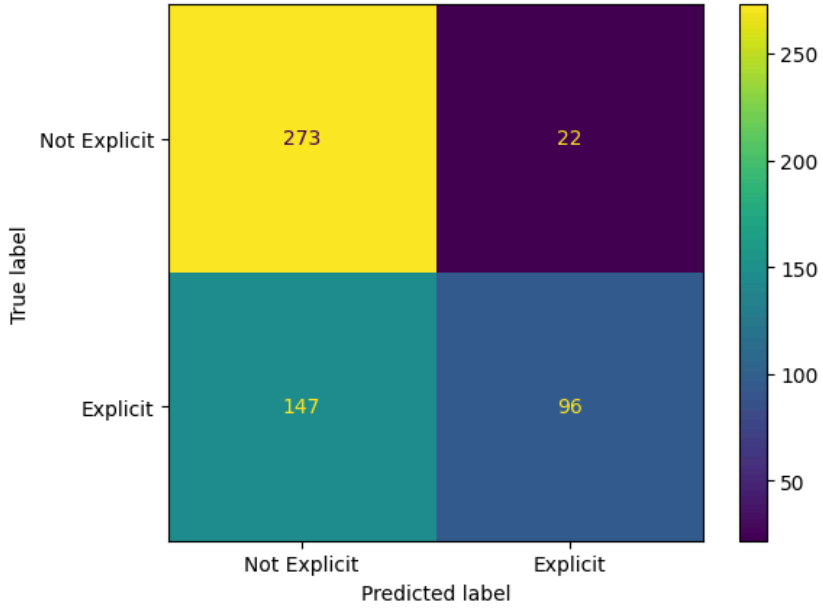
Analysis technique

Once we had our data prepared and loaded into a CSV file, we went through each lyric and split it into individual words, then found and removed each stop word (described further in the Technical section). We then used a word stemmer tool to remove suffixes and prefixes and leave a more standardized dataset for our model to work with. We then defined our model's input features (stemmed lyrics) and output features (explicit rating) and created a function for our Naive Bayes classifier. We used the train_test_split function from scikit-learn to randomly choose training and testing groups and used the same library to create a classifier. We trained our model on the appropriate subset of data and used it to predict our test subset; by comparing that prediction to the actual explicit ratings of the songs in that subset, we were able to determine the strength of our model.

Results



When we ran our model as is on all features (counts of every word in all the song lyrics), we were able to achieve the above confusion matrix. When we ran a Monte-Carlo cross-validation, we found that our model has on average a precision of **0.901**, a recall of **0.827**, an f-score of **0.862**, and support of **234.7**.



We then trained the classifier using only the 30 most informative features of the first classifier. When we used Monte-Carlo cross validation with this model, we got on average a precision metric of **0.828**, recall of **0.423**, f-score of **0.559**, and support of **239.6**. This low recall score comes from the fact that we had quite a few more false negatives (negative representing an explicit song in this case) and far fewer true negatives. Prediction of explicit songs maintained a similar level of accuracy to the first model.

Technical

One of the most important things we did in cleaning our data was using a natural language library (nltk, which stands for natural language toolkit) to remove stop words like “the”, “and”, and “is”; these are words that contribute to the grammatical sense of a text but don’t themselves communicate any important information. Removing these allows us to process our data more quickly and ensure that unimportant words aren’t causing less accurate results. We also used nltk’s PorterStemmer tool to remove morphological affixes from words; this standardizes our words and allows more patterns to be recognized.

We believe that a Naive Bayes classification model would be perfect for our purposes as it specializes in predicting a label based on a block of text. Since we are just looking to distinguish between the “explicit” and “non-explicit” labels, this model is perfect for our needs. In the process of setting up this model, we were very interested in finding out which words informed our model the most as this would tell us whether our model was determining based on swear words (which we expected) or whether it was using other information in its decision. This could give us some fascinating insight into the attributes that explicit songs have in common and potentially reveal common similarities between non-explicit songs. Unfortunately, this is not something that scikit-learn supports and we were not able to retrieve this information.