Artifical Intelligence 21F Final Project

Tina Buzanis[1]

December 18, 2021

About the dataset

This dataset contains 13 audio features, along with the genre, for each of 11,741 songs. These songs were collected using Spotify's API, and organized in an sqlite database, which is included with this file. There was, however, a unique challenge in that Spotify does not tag their songs with a genre. Rather than manually assigning genres to each of the 11,741 songs, I instead chose the songs based on genre: for each of the genres I chose to include (chill, indie/alt, edm/dance, jazz, rock, metal, pop, and hiphop), I downloaded the top playlists available within that genre on Spotify. This gave me a roughly equal distribution of samples from each genre. From there, I was able to proceed to the exploratory data analysis.

I. EDA

Each of the 11,741 songs in this dataset belong to one or more genres, though the distribution is roughly equal:

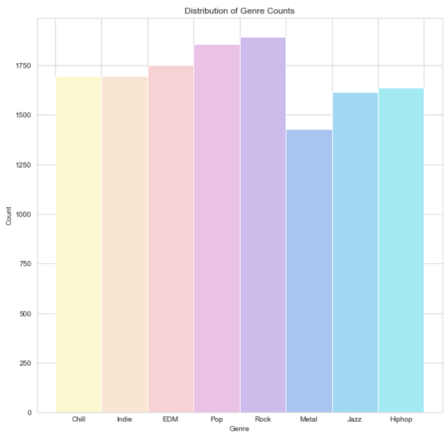

Figure 1: Genre Distribution

However, despite the distribution being roughly equal, there was a unique challenge in this data set in that there is not much correlation between the features:

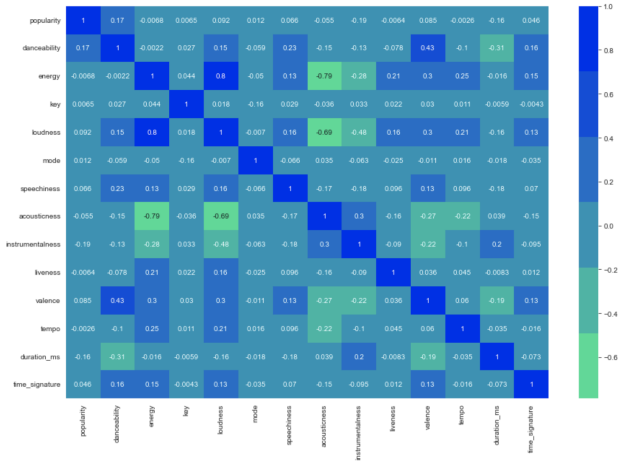

Figure 2: Heatmap of Genre Features

Through continued trial, error, and inspection of the features' relationships, I narrowed the list of features to be used down to seven: popularity, danceability, energy, key, valence, tempo, acousticness, and instrumentalness. Spotify delivers the audio data prescaled. That is, the features directly relating to the audio are of a continuous nature, scaled already to be between zero and one. Features that do not directly correspond to the song auio, such as popularity, key, and tempo, are not scaled. To fix that, I applied both a StandardScaler and a MinMaxScaler to all features that were not standardized. This brought all of the data features onto the same scale, and around a similar distribution.

II. Naive Bayes - Gaussian

The first model I used was a Gaussian Naive Bayes model. As one may infer from the name, Gaussian Naive Bayes models perform most accurately on data which follows a Gaussian distribution. It was for this reason that I applied the scalers udring the EDA. The confusion matrix and F1 score for both the training and test sets is shown below.

Surprisingly, running the model on unscaled data did not provide much of a difference - about one percent improvement for both the training and test cases. This may be because even scaling will not make cor-

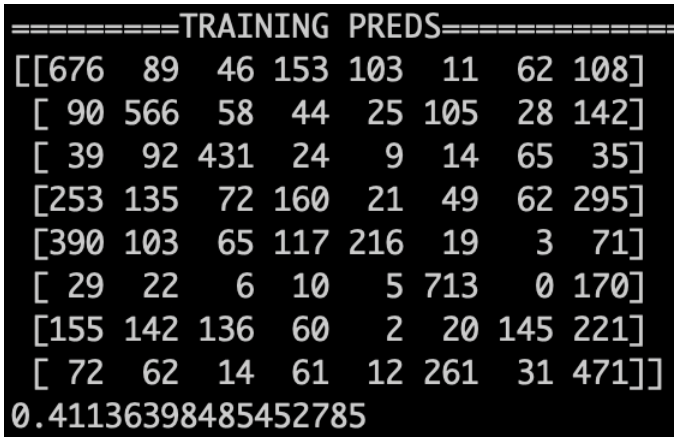

Figure 3: Training

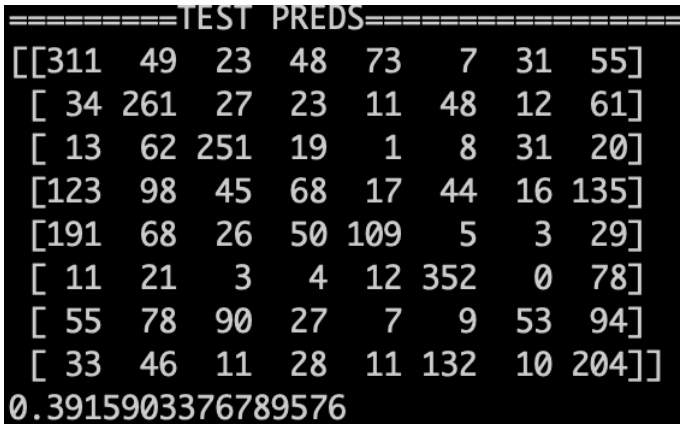

Figure 4: Test

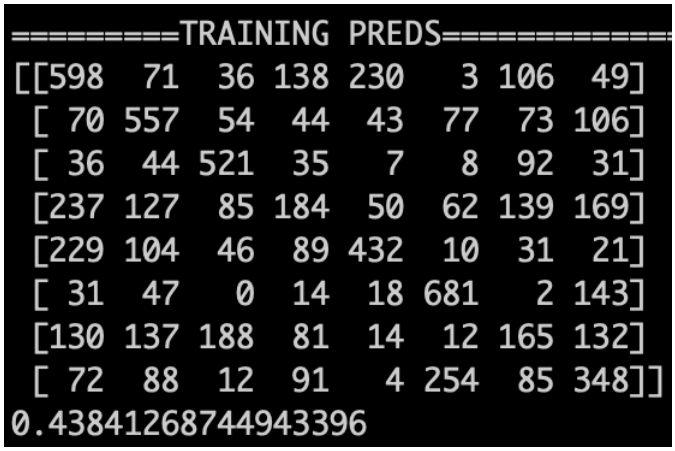

Figure 5: Training

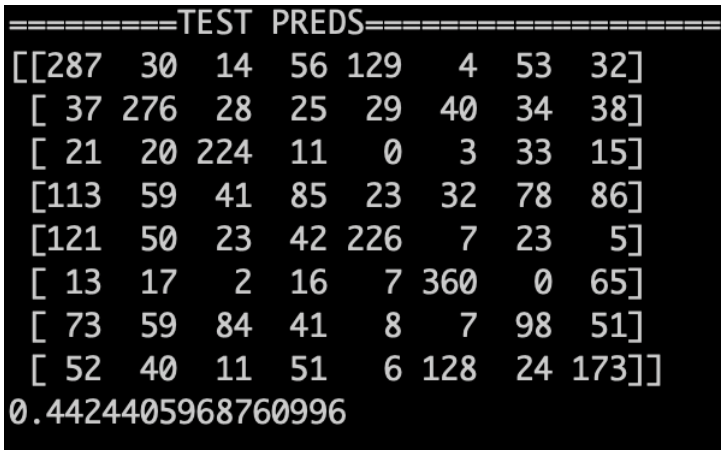

Figure 6: Test

relations appear where there are none.

III Naive Bayes - Categorical

The second model tried used a Categorical Naive Bayes algorithm. This classifier works best on data sets that are discrete, or categorical, in nature. This data set has a mixture of categorical and continuous features. As was done for the Gaussian classifier, all features were made categorical prior to feeding the data to the model. This model performed similarly to the Multiclass Logistic Regression model - slightly better than the Naive Bayes. This is not surprising, considering that a good part of the data from this dataset is categorical in nature. Given more time, I would go and carefully formulate how I would discretize the data, instead of doing it all at once. I would also like to explore the different effects different ways of filtering affect the results. The confusion matrices for the training and test predictions are below:

IV Multiclass Logistic Regression

Next, a multiclass logistic regression model was tried. As Regression models favor continuous data, I once again scaled and centered the data prior to feeding it to this model. This algoritm performed about as well as the Gaussian Naive Bayes, with about a 43% accuracy.

V. Neural Network

The last algorithm evaluated on this data set was a Neural Network. Having implemented the algorithm from scratch, I did not have access to the same metrics as I would through sklearn. However, after implementing a model with six hidden layers, and running it over 1000 epochs with a learning rate of 0.005, the results appeared to be the best out of the models so far:

Although the model had a difficult time classifying

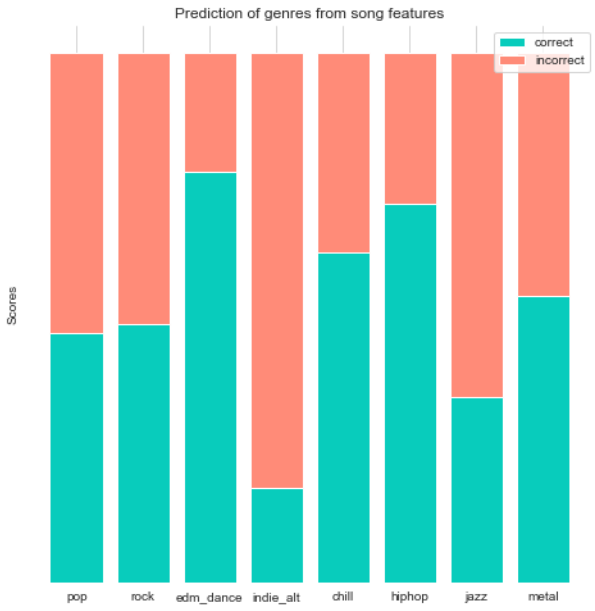

Figure 7: Neural Network

indie-alternative songs, it did well with edm-dance, chill, and hiphop. A deeper exploration of the data is necessary, as is the time to tune the hyperparameters of this model. Several iterations were more ”balanced” in that the number of correct predictions were lower, but more evenly distributed across the genres. However, in seeing that this model is quite successful at classifying several genres, it is worth a deeper look into the data to see the trends the algorithm may be capitalizing on.

VI. Conclusion

Although the performance of these models is not perfect, their performance is acceptable given the size and nature of the dataset. One difficulty that was present was that running the algorithms on the full dataset took a rather long time - about thirty minutes each run. That made the process of iterative improvement time-consuming, as I had to wait a while after making a change to see the results. From the data collected, it appears that the Neural Network performs the best, although it likely presents with a similar accuracy to the Naive Bayes and Logistic Regression models on average. Given more time, I would spend more time exploring the data, and engineering new features. I would also explore different genres, or perhaps a different nuber of genres. Additionally, I would compare the results from the algorithms I wrote to sklearn’s implementation, and see which version performs better on this dataset.