Classifying Genres in Spotify Songs

Timothy Nguyen

Johnson Nguyen

Abstract— Created a K-Nearest Neighbors Classifier to approximate the genre of a selected song from Spotify. It was found that a KNN classifier was effective in predicting a song's genre with a higher than 70% accuracy

I. Introduction

In the project, we sought out to create a model that would accurately and consistently classify songs according to a genre that it belongs in. In this project, four types of song genres were observed: K-Pop, EDM, Hip hop, and Jazz. The primary goal of this experiment is to evaluate whether or not a K-Nearest Neighbor classification is sufficient enough to predict song genres.

II. Method

K-Nearest Neighbors

In this classification process, a K-nearest neighbor method was used to predict the genre of each individual song. The goal is to create neighborhoods for 8 features of each song and compare each song's features' distances with its nearest neighbor.

III. Experiment

In forming the data, four different playlists were created according to the four song genres that were observed. Each playlist consisted of 100 songs each all belonging to the one specific genre.

A. Data Collection

Each song was a part of a Spotify generated playlist that was related to its respective genre. For example, a playlist was used for Jazz in which all the songs were of the Jazz genre.

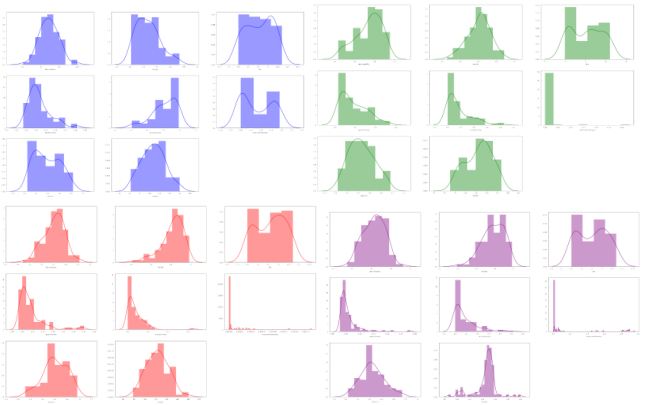
In order to use these playlists, a Spotify package called Spotipy was used in order to pull the playlists' contents into a usable medium. Each playlist had its own special ID that, when used with Spotipy, would return the contents of said playlist.

Using each playlists' metadata, songs in the playlist would have to be retrieved according to their song IDs. Since calling the method to return a playlist would return a list of song IDs in each playlist, an array would be made for each playlist containing only the song IDs.

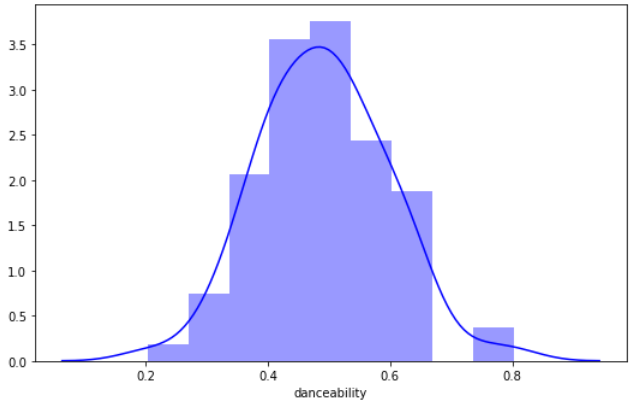
After creating the four individual arrays of Song IDs, a Spotipy method would be made to then return each song's metadata, creating a subarray for each song in the genre arrays. Once this was created, the data can now be imported into a datafile in Excel format.

A song's metadata consisted of various values that were specific to it. These values were the basis of comparison for our experiment. The values that were observed were:

- Danceability - Measure of how suitable a track is for dancing.
- Energy - Measure of intensity and activity.
- Key - Estimated overall pitch of the track.
- Speechiness - Measure of likeliness there will be vocals.
- Acousticness - The likelihood that the song was performed in person.
- Instrumentalness - The likelihood the song will not have any vocals and only instrumentals.
- Valence - Measure of how happy the track sounds.
- Tempo - Measure in beats per minute of how fast a song plays.



Creating four different Excel sheets, each song's features were aggregated and can now be observed in the form of a bar graph. As seen above, blue is Jazz, green is Hip Hop, red is K-pop, and purple is EDM.



An example of these graphs can be seen in the Jazz Excel file's Danceability feature.

What was later observed was that the features of "Key" and "Tempo" consisted of values outside the range of [0,1]. The other 6 features' observations were within that said range. So in order to reduce bias in the data, a reduced scaling of these two features must be made. By using the equation:

$$x_{new} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

"Key" and "Tempo" values are now scaled down to a range of [0, 1] ensuring an even bias between all features.

After scaling these features, all the observations in each datafile were then given a "Class" feature with a value of 0, 1, 2, or 3 corresponding to if the song is Jazz, Hip hop, Kpop, or EDM respectively. The datafiles are then merged into one datafile called full_songs and the analysis can now be done.

B. Train and Test

Running an initial train_test_split, full_songs is split with 80% being a training set and 20% being a testing set. Using this split, a KNeighborsClassifier is made using the Scikit learning package.

By fitting the KNN classifier using the training set, it can then be used to predict the target variable (Class) in the testing set through the k neighbors that were created.

In K Nearest Neighbors, the distances of each feature to a k neighbor would be compared. Using the Euclidean distance algorithm,

$$d(x, x') = \sqrt{(x_1 - x_1')^2 + \ldots + (x_n - x_n')^2}$$

a comparison of each feature can be made and whatever k neighbor the features' distances are closest to would be that corresponding genre class.

Using this knowledge, a prediction can be made using the KNN classifier on the testing set. Doing this method returns an accuracy value, which determines how accurate the predictions were in comparison to its real values. In this case, the accuracy was found to be

```
Accuracy is:
0.7763157894736842
```

C. Cross Validation and Optimization

The process of cross validation is to ensure that across all iterations of the train test split, the accuracy of the predictions are constant. In this case, two cross validation methods were used, a standard cross validation and a grid search cross validation. In both cases, the scores of both methods were above 70% which is an acceptable amount.

In a standard cross validation, a fold amount of 5 means that it would run the train test split with 5 different iterations of training sets and test sets. So, it would run train test split 5 times with different sets. It would then compare each accuracy score and display them. In this case, the scores were:

```
Running Cross validation score...
[0.75409836 0.7704918  0.70491803 0.72131148 0.83333333]
```

In this, it can be observed that all instances were higher than a 0.7 score with the highest being 0.8333.

In a Grid Search, the algorithm returns the best parameters for a model such that it would perform with optimal results. In the case of the experiment, a grid search was run for a KNN classifier and returned the parameters that can be run in KNN that would give the most desired results.

```
The best score is:
0.7342105263157894
The best parameters are:
{'n_neighbors': 4}
```

In this case, the best score is a 0.7342 and the best parameters for a KNN classifier of this dataset was 4, which supported our initial number of 4 genres.

RESULTS

After running our analysis and reporting our findings, it was found that a K Nearest Neighbors Classifier was an effective use of classifying songs by their genres. With accuracy scores of 70% and above for all tests done, it can be seen that it is a reliable method.

An explanation on why it is not a near perfect accuracy score can be seen as that because we have 4 different genres, there may be too far of a spread for the classification to totally confirm what genre it is. That is, if one were to refer to the series of bar graphs for each feature in each genre class, it can be observed that some genres (i.e Hip Hop and K Pop) exhibit similar feature values. Because of this, there may be a few songs that are misclassified. A way around this can be to find a way to include a feature that would include language. That way, it can be easier to decipher from those close cases.

However, the end results have consistently shown that a KNN classifier successfully correlates songs to their respective genres.

REFERENCES

[1] 3.1. Cross-validation: Evaluating estimator performance. (n.d.). Retrieved December 17, 2020, from https://scikit-learn.org/stable/modules/cross_validation.html

[2] Sklearn.model_selection.GridSearchCV. (n.d.). Retrieved December 17, 2020, from https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html