Machine Learning

Week 1 Assignment

Overview

You will build a simple *k Nearest Neighbors* machine learning classifier than can differentiate between 2 genres of songs.

User Stories

You will complete the following requirements:

1) Create a *balanced* dataframe of songs, containing only 2 *consolidated* genres. You can choose any two genres from the following list:

```
* `pop`
* `dance`
* `r&b`
* `rap`
* `country`
* `rock`
* `jazz`
```

2) Produce a kNN Classifier that has a precision, recall, and an F1 score all upward of 90% on a single `train_test_split` fold of data. Show the confusion matrix and classification report proving this score.

3) Show all cross-validation scores of the classifier, including the average score. The average score should be upward of 90% as well.

The following advanced user stories are optional. You're not required to do these, but you will learn more from doing them:

- Visually demonstrate the relationship between different values of *k* (which is simply the value of `n_neighbors` that you pass to your `KNearestNeighborsClassifier` object), and the mean cross-validated F1 scores.
- Experiment with different pairs of genres, and see which 2 genres are most easily differentiable by your classifier.
- Use unbalanced data and compare your results with balanced data.
- Build a classifier that works on more than just 2 genres, while still maintaining a 90%+ accuracy/precision/recall rate.

Hints / Walkthrough

1) Read the provided JSON database of songs, `MasterSongList.json`, into a pandas dataframe using `.read_json`. You will see that one of the entries in the dataframe is a list, called `audio_features`. This will be the feature vector data you will eventually have to use in your classifier. There are 17 audio features in a single vector, and they correspond to the following musical parameters:

```
1. `key`
2. `energy`
3. `liveliness`
4. `tempo`
5. `speechiness`
6. `acousticness`
7. `instrumentalness`
8. `time_signature`
9. `duration`
10. `loudness`
11. `valence`
12. `danceability`
13. `mode`
14. `time_signature_confidence`
15. `tempo_confidence`
16. `key_confidence`
17. `mode_confidence`
```

2) Extract and consolidate the genres in the dataframe to only the following (see the **Tips and Notes** section below for more information on how to do this):

```
* `pop`
* `dance`
* `r&b`
* `rap`
* `country`
* `rock`
* `jazz`
```

3) Build a balanced database of songs using only 2 genres chosen from the list above (see the **Tips and Notes** section below for more information on how to do this).

4) Build an audio feature matrix with the `audio_features`, and the 2 consolidated genres of your choice as the classification labels.

5) Construct a kNN classifier and fit it with the given features and labels. Run a 30% `test_train_split` and generate the confusion matrix and classification report.

6) Run a 10-fold cross-validation on the classifier, showing all scores, and calculating the mean as well. Tweak the classifier parameters if necessary to get an accuracy/precision/recall of at least 90%.

Tips & Notes

- **Consolidating Genres**
  Certain song genres are in the format `genre: sub-genre`. One example is `rock: contemporary alternative`, and `rock: classic rock`. Consolidating genres simply

means that both `rock: contemporary alternative` and `rock: classic rock` are simply treated as `rock`. You can use functions in pandas to iterate over all the entries and replace those genres containing sub-genres, with just the genre (i.e. replace `rock: contemporary alternative`, `rock: classic rock`, etc. with `rock`). The `.loc`, `.map`, `.split`, `.apply`, and `lambda` functions can help you do so.

- **Balancing Data**
  Balancing your dataset simply means that you ensure there are an equal number of samples for both your classes when training your classifier. For example, let's say you pick `rock` and `jazz` as your 2 genres, and there end up being 10,000 rock songs but only 2,000 jazz songs in your database. When constructing your feature matrix for your classifier, make sure that you use an equal number of songs for both -- 2,000 `rock` songs and 2,000 `jazz` songs (you can either pick the first 2,000 `rock` songs, or a random subset -- see *Extra Credit* below). We're doing this to ensure our classifier is equally 'trained' on both sets of data, and hence treats both genres equally well (or poorly) — for now.

- **Tweaking Parameters**
  Some parameters that you can try tweaking to play with the accuracy/precision/recall metrics are the number of neighbors, the algorithm used to search for the neighbors, and if you're feeling adventerous, you can also try the following:

  - Randomize the subset of songs for the genre with fewer samples. In our `rock` and `jazz` example, when selecting 2,000 `rock` songs out of 10,000, select a random subset instead of selecting the first 2,000. Experiment with the randomization of this subset and see if/how it affects your classifier metrics.
  - Experiment with different genres and see if/how it affects your classifier metrics.