Week 2 Assignment

Overview

You will implement 3 new kinds of classifiers and programmatically find out the best parameters and features to use for them.

User Stories

You will complete the following requirements:

1) Produce a balanced dataset, consolidated to the following genres only:

- `jazz`
- `dance`
- `rock`
- `rap`

2) Build the following classifiers:

- Logistic Regression
- Support Vector Machine
- Random Forest

3) Successfully find the best values for the following classifier parameters using `GridSearchCV`:

Logistic RegressionSupport Vector MachineRandom Forest `solver` `multiclass` `C` `gamma` `n_estimators` `min_samples_split` `max_features`

4) Successfully find the best `audio_features` for all classifiers using the following feature selection methods:

Logistic RegressionSupport Vector MachineRandom Forest `SelectFromModel` `RFE` `SelectKBest` `SelectFromModel` `RFE`

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

- Try using and analyzing `moods` as a feature as well. Is it more important or less important than `audio_features` when trying to predict `genres`?
- Experiment with different values for the following:
  - `k` for `SelectKBest`
  - `threshold` for `SelectFromModel`
  - `n_features_to_select` for `RFE`
- Do the above using `GridSearchCV`! You will have to use a `Pipeline` object in python to combine both parameter selection and feature selection, such that a single `GridSearchCV` instance can work on both processes. So:
  - For the Support Vector Machine, one instance of `GridSearchCV` will try different `k` values for `SelectKBest`, while *also*trying different `C` and `gamma` values for your `SVC` estimator.
  - For Logistic Regression, one instance of `GridSearchCV` will try different `solver` and `multiclass` values for your `LogisticRegression` estimators, while *also* trying different `threshold` values for `SelectForModel` / `n_features_to_select` values for `RFE`.
  - For Random Forest Classification, one instance of `GridSearchCV` will try different `n_estimators`, `min_samples_split`,

and `max_features` values for your `RandomForestClassifier` estimators, while *also* trying different `threshold` values for `SelectForModel` / `n_features_to_select` values for `RFE`.

Hints / Walkthrough

1) Last week, you worked with 2 genres of your choice. This week, we will work with 4:

- `jazz`
- `dance`
- `rock`
- `rap`

In addition, there are a few more rules to consolidate them:

1. Make sure that there are `1500` songs for each genre (so you will have a total of 6000 songs)
2. Only select songs that have `yt_views > 1000`. [**Note**: `yt_views` represents the number of YouTube views that the song has. So basically the idea is to build a balanced dataset of 'popular' songs]

2) Once you have your dataset, build your 3 classifiers, and make sure to use the following values for each of them:

Logistic RegressionSupport Vector MachineRandom Forest `solver='saga' multiclass='multinomial' C=1 gamma=1 n_estimators=5 min_samples_split=2 max_features='log2'`

3) These will be your *original* classifiers. For each, show the `confusion_matrix` and the `classification_report` on a `train_test_split` of `0.3`. Then, use `GridSearchCV` to try the following values:

Logistic RegressionSupport Vector MachineRandom Forest `solver: ['newton-cg', 'sag', 'saga', 'lbfgs'] multiclass: ['ovr', 'multinomial'] C: [0.1,1, 10] gamma: [1,0.1,0.01,0.001] n_estimators: [5, 10, 100] min_samples_split: [2, 3, 4, 5, 10] max_features: ['sqrt', 'log2', 'auto']`

3) The results of your `GridSearchCV` will show you the best parameters for your classifier. Build a new classifier using these parameters. This is your *best* classifier. Show the `confusion_matrix` and the `classification_report` on a `train_test_split` of `0.3`.

- Compare the results of your *best* classifier to your *original* classifier and discuss your results. What are your observations? Does your data make sense?

4) Now, the final step is to analyze which `audio_features` seem to be most important. Choose a number of features (see **Tips & Notes**) and use the methods outlined in Story # 4 to find out which features are most important.

- Print out the names of the selected features, and show their ranking / score / importance.
- Re-fit your *best* classifier on this new, reduced feature set, and re-produce the `confusion_matrix` and `classification_report`. This is your *optimized* classifier.
- Compare your *optimized* classifier to your *original* and your *best* and discuss your results. What do you observe? Does it make sense?

Tips & Notes

- **Finding the best features**Remember that the `SelectFromModel` and `RFE` methods only work on those classifiers that have a `feature_importances_` or `coef_` attribute. Support Vector Machines don't expose this attribute, which is why we try the `SelectKBest` method on it instead.
- **Choosing a reduced number of features**For the Logistic Regression and Random Forest Classifier, try seeing how many features are returned by `SelectFromModel`, and then use that number for `RFE` as well. You can also use that number for the Support Vector Machine, or feel free to use a number of your choice (egs: **5** features). Try to explain your choice. Remember to show the names, as well as the scores / rank / importance of the features that you end up selecting!
- **Trying different values independently**Some of you might notice that we are varying the optimal parameters for our classifier *independently* of the number of features we are trying to reduce to. Technically, this is not 'perfect', because there can sometimes be interactions between the parameter values we choose in one step and the optimal value for a downstream step. In other words, to avoid local optima, we should try all the combinations of parameters, and not just vary them independently. This can be achieved if you try the bonus story, which involves using a `Pipeline` object.
- **Don't worry about getting a high score!**The goal of this assignment is not necessarily to try and get a maximum (or even improved) score, but to instead get comfortable with using different techniques to see how you can play with different features. The idea is that, once you know how to do so, you can then use these techniques in the future when building your own machine learning classifiers to improve them.