Classification challenge

Machine Learning with Applications in Python

INF 2179 – Winter 2024

Introduction: The goal of this classification challenge is to build a classifier that can accurately predict the star rating of recipe reviews (sentiment analysis)

General information:

- Maximum number of points: 30 (account for 30% of the semester)
- Deadline: March 18th (to be submitted to Quercus before 23:59 PM)
- Individual challenge (group work not permitted)

Dataset:

Your objective is to develop a machine learning model that can predict the star rating of different recipe reviews on a 1 to 5 star rating scale with **a score of 0 denoting an absence of rating**. Two CSV datasets (adapted from https://www.kaggle.com/datasets/joebeachcapital/recipe-reviews-and-user-feedback-dataset) are given to you named train.csv and test.csv.

It is important to note that the model should only be trained using the "train.csv" dataset and evaluated using the "test.csv" dataset.

Hints:

Below are the ***suggested*** steps to take:

1) Extract some features from the reviews using:
    a. CountVectorizer from scikit (play with the various parameters!)
    b. Extract some basic text information using pandas (e.g., length of the text)
    c. Feel free to use any other feature extraction techniques (using external python libraries is allowed). You may not use all the features in the dataset.
2) Ensure that you inspect the data and apply any data cleaning needed.
3) Try various classifiers and various parameters. Here are some classifiers you can try:
    a. Decision trees and random forests (with various depths and features)
    b. Naïve Bayes
    c. Feel free to try any other classifiers not seen in class (e.g., classifiers in scikit). Note that it is possible to get the maximum number of points for the accuracy using approach A or B (if the features extraction is done well).
4) Try to do a grid search on hyper parameters of your classifier
5) Once you have selected the best classifier, use the training set to build a final classifier.

Instructions for submission:
A valid submission should be a zip containing **exactly 3 files** :

1) pred.csv: a file containing 3637 lines one line per test prediction (do not include the index nor the header, use index=False, header=False when saving the column). The file should look like the following:
   pred.csv:
   1
   2
   5
   .
   .
   .

2) accuracy.csv: a file containing the accuracy (on the test data) include neither the index nor the header, see below:

   accuracy.csv:
   0.76

3) Notebook file: the file used to generate the model and **all the above files, plus the answer to the questions (see below).**

After preparing these three files, you should upload them to Quercus as a ZIP file **named with your student ID (e.g., 1007111023.zip).**

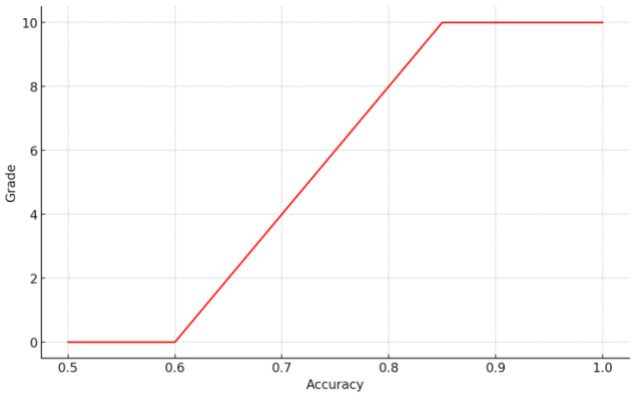
The code below shows how the accuracy and predictions can be exported in the right format.

```
pred = classifier.predict(test_set)
pred = pd.Series(pred).to_csv('pred.csv', index=False, header=False)
```

Evaluation:
The submission will be evaluated using three criteria:

1) Accuracy on test data (10 points)

$$\text{Formula} = \min\left(\max\left(\frac{\{\text{accuracy}\} - 0.6}{(0.85 - 0.6)}, 0\right), 1\right) * 10$$



2) Code (10 points)
    a. The code will be manually inspected and evaluated. To get the maximum number of points, the jupyter notebook must:
        i. Be compact
        ii. Contain some comments (only where necessary)
        iii. Be efficient (run in a reasonable amount of time, unless an advanced data analytic is being used)
        iv. Easy to run (should be run in a matter of a few minutes, if you do grid search, make sure to comment out those sections and include the results only)
        v. **NOTE:** DO NOT delete/clear the Notebook's outputs!

3) Questions (10 points): Please respond to the following questions **within your code notebook near the end of it.** Incorporate the **questions** and **answers** as text elements (check here for a tutorial on how to do it). If any question requires computation, such as calculating precision or recall, ensure to execute the necessary code within the notebook and include both the code and its output.

    a. How does the performance of your model vary across different classes? Analyze and discuss your observations regarding the precision and recall metrics for each class.
    b. Considering your analysis, how would you recommend using this model in a real-world application? Discuss any limitations or considerations that should be taken into account.
    c. Analyze your data to address the previously identified accuracy issues. Describe your method to address this issue, implement it in code and retrain a classifier, and assess any improvements or ongoing challenges. Your evaluation will be based on your method's appropriateness, not the results.

Important notes:

- Group work is not permitted. Students having highly similar codes risk having their work rejected (0/30).
- The work should be reproducible, i.e., using your jupyter notebook, it should be possible and easy to reproduce the same predictions. **If this is not the case, the work will not be accepted!** To achieve this, don't forget to fix the random state (random_state parameter). I suggest that you try to re-run the entire notebook to ensure the predictions remain identical.
- Data augmentation is not allowed. You cannot add additional information (from any other dataset). However, using existing libraries to extract new features is allowed (data augmentation != feature extraction). If you are unsure, do not hesitate to ask me.
- The data is provided "as-is". If you have a personal issue working with such data, please do not hesitate to let me know by email **before the March 13th.**

As always, if something is not clear, please do not hesitate to ask on the discussion board.