Mini-project 2

This mini-project is to be completed in groups of up to three. You will submit your assignment on myCourses as a group. You must register your group on MyCourses.

Introduction

This mini-project will give you experience with working with text data and machine learning with scikit-learn. You will develop models to analyze text from the website Reddit. Reddit is a popular social media forum where users post and comment on content in different themed communities, or subreddits. The goal is to, given a post or comment from reddit, identify the subreddit where that post or comment originally came from. The dataset is gathered from posts and comments from 8 subreddits:

1. rpg : Subreddit for tabletop role playing games
2. anime : Subreddit for discussing Japanese animation
3. datascience : Subreddit for discussing matters related to data science, including machine learning
4. hardware : Computer hardware news and discussions
5. cars : Discussions and news about cars
6. gamernews : Discussing video game related news.
7. gamedev : Subreddit for video game developers
8. computers : Subreddit for discussing anything about computers.

Download the training and test data from the Kaggle competition page (details to follow below). Each line in train.csv includes two fields: text of a post/comment, and the subreddit it was posted on. Use train.csv for training your models.

Your tasks

Part 1: using the data provided in train.csv

1. Implement Bernoulli Naïve Bayes from scratch. Since your implementation may not be optimized for speed, you can set the max_features=5000 in CountVectorizer(). This will consider only the top 5000 words i.e. m=5000.
2. You need to do additional experiments using at least one additional classifiers from the SciKit learn package (excluding Bernoulli Naïve Bayes). For example Logistic Regression, Decision Tree, etc.
3. Implement a model validation (e.g. using k-fold cross-validation) and use it to report, in your write-up, the performance of the above mentioned classification models (i.e., your Naïve Bayes model and the SciKit learn model(s)). You can use KFold from SciKit learn.

Part 2: run your best model (obtained in Part 1) on the data provided in test.csv and submit the result on Kaggle competition. (See below for more details)
Part 3: Prepare and submit your report (max 5 pages) along with your codes (details below).

Hint 1: You many want to use Laplace smoothing with your

Bernoulli Naïve Bayes model.
Hint 2: you may use CountVectorizer() with binary=False to obtain a binary vector representation for a text.
Hint 3: when debugging your code, you might start off by using only a quarter of train samples and a quarter test samples. When you are done with code debugging, apply your code to the complete data.

Kaggle competition

test.csv contains the test samples. Each row in test.csv contains review id and the text of the review. You need to make a prediction for every test sample and submit a .csv file in Kaggle where each line contains review id and its predicted label. ExampleSubmissionRandom.csv is an example of the kind of csv file you should submit in Kaggle. Note: in the example file, the class label written in front of each id is just a random value and you should replace them with your predicted values. You are limited to two submissions per day.
You must register in the Kaggle competition using your McGill email address. If you already have a Kaggle account under a different email address, do not delete your account. You only need to change your email in your Kaggle profile. You must form a team in the competition page on Kaggle, the name of your team must be the same as the name of your group on myCourses (e.g. Team Name: Group 2). Except where explicitly noted, you are free to use any Python library for this project. You are not allowed to use any training data other than what is

provided for the competition. Here is the **link to the Kaggle competition**.

Report

We are flexible on how you report your results, but you must adhere to the following structure:

- Abstract (100-250 words): provide a summary of the project task and highlight your most important findings.
- Introduction (at least one paragraph): Summarize the project task, the dataset and your important findings. This is similar to the abstract but you should provide more details.
- Dataset (at least one paragraph): Briefly describe the dataset and its characteristics. Also, describe the preprocessing steps for preparing the feature vectors. Note: You do not need to explicitly verify that the data satisfies the i.i.d. assumption (or any of the other formal assumptions for linear classification).
- Proposed Approach: Briefly describe the features you designed and the methods you have implemented or used for this project. No need to provide detailed derivations and proofs but you should provide some (a few sentences) background, description and motivation for each model. You should properly cite and acknowledge previous works/publications that you use or build upon. Discuss any decision about training/validation splits, algorithm selection, regularization, hyper-parameters, etc.
- Results: Summarize your results using tables and/or figures. Discuss the results for each model (for example accuracy, runtime, etc.). Since you do not have the labels for the test set,

your results should be based on your validation set(s). Report your test set leaderboard accuracy too.

- Discussion and conclusion: Discuss and summarize the key takeaways from the project and possible directions for future investigation.
- Statement of Contributions (1-3 sentences) State the breakdown of the workload across the team members.
- Appendix To facilitate the grading process, attach the codes for your implementations to the end of your report. This does *not* count towards the page limit of the report. You must also submit your code as a .zip file.

  You are expected to discuss your findings with scientific rigour. you can discuss why did you get the results you did? Compare and contrast the different algorithms. What sort of changes might you make to each of those algorithms to improve performance? How fast were they in terms of wall clock time? Iterations? Would cross validation help? Which algorithm performed best? How do you define best? Be creative and think of as many questions you can, and as many answers as you can. Please keep your analysis as short as possible while still covering the requirements of the assignment. The analysis report is limited to 5 pages (single-spaced, minimum font size of 10 and 1 inch minimum margin each side). We highly recommend to use LaTeX for preparing your report. We recommend NeurIPS2020 LaTeX template. Your report should look technical. Imagine you are writing a paper for a major machine learning conference.

Deliverables

- report.pdf: Your report (including Appendix) as a single pdf file. This must be submitted as a separate file.
- code.zip: Your codes (e.g. .py, .ipynb, etc.) must work with Python 3.6 in Colab. Include a *readme* file and provide instruction for TA on how to replicate your results on Colab. All the results must be reproducible in Colab using the submitted code.zip. Points will be deducted if we have a hard time reading or understanding the structure of your code. Do *not* put the report.pdf in the zip file.

  The report should be self contained. TA's will do the grading mainly based on the report.pdf, and will not be obliged to consult the supplementary codes.

Evaluation

This is an open-ended project. Feel free to go beyond the minimal requirements. The evaluation has two parts each worth 50 points out of 100.

Performance (50 points): This is based on the performance of your best model on the held-out test set on the Kaggle competition. Your grade will be computed based on a linear interpolation between three points: the 2nd top group, a TA baseline and a random baseline. The random baseline is the score needed to get more than 0% on the competition. The TA baseline is the score needed to get 75% on the competition. In other words, if your score is between the random and TA

baseline, your grade is a linear interpolation between 0% and 75% on the competition; likewise, if your score is between the TA baseline and the 2nd best group, your grade will be between 75% and 100% on the competition. In addition to the above criteria, the top two groups all receive 100%. Additionally, the top group will receive 10% points as bonus.

Quality of your report and proposed methodology (50 points): Your report should be both thorough and concise. It will be judged based on its scientific quality including but not limited to: Does the report include all the required experiments? Is the report technically sound? How thorough/rigorous are your experiments? Is the report well-organized and coherent? Is the report clear and free of grammatical errors and typos? Does the report contain sufficient and appropriate references and related work?

All members of a group will receive the same mark.

Final remarks

You are expected to display initiative, creativity, scientific rigour and critical thinking skills. You don't need to restrict yourself to the requirements listed above - feel free to go beyond, and explore further.

You can discuss methods and technical issues with members of other teams, but you cannot share any code or data publicly or with other teams. Any team found to cheat (e.g. use external information, use resources without proper references) on either the code, predictions or written report will receive a score of 0

for all components of the project. Sharing or posting the mini-project specifications, including your code, publicly, whether intentionally or unintentionally, is considered academic misconduct