Fall 2023
CSCE 320 - Principles of Data Science
Homework #3

Problem 1 – Classification tree (35%)

The dataset 'titanic.csv' included in the assignment contains information about passengers onboard RMS Titanic on its maiden voyage. Your goal is to build a classification tree to predict which passengers survived based on the provided information.

- **a. 15% credit.** Grow a classification tree on the entire training set. Visualize the tree (e.g., as in L11, slide 24), discuss which variables are selected at each split and explain the meaning of those decisions, analyze node purity, etc. This part of the exercise is open-ended. Explore the model and see what you can find out.
- **b. 10% credit.** Repeat the previous process, but this time using different values for the pruning hyperparameter α (see L11, slide 20). Analyze and discuss the results.
- **c. 10% credit.** Estimate classification performance using 10-fold cross validation. Discuss which pruning criteria you used and how it impacted classification performance on unseen data.

Problem 2 – Ensemble learning (35%)

Using the nutrition dataset included in HW2 / Problem 3, develop a classification model based on bagged classification trees.

- **a. 15% credit.** Train a bagged tree model with 50 trees using the training data (x1). Estimate OOB performance as a function of the number of trees grown. Discuss your findings.
- **b. 10 % credit.** Generate a variable importance plot for the model you developed in part (a). Interpret the results –this will require you to do some research on the meaning of the various predictors for different food categories.
- **c. 10% credit.** Train a sequence of bagged tree models with increasing numbers of trees (from 1 to 50). Estimate model performance on the test set (x2). Compare your results against the OOB results in part (a). Discuss your findings.

Problem 3 – Bag of words (30%)

Included in the assignment are four CSV files containing information about movie reviews:

- reviews.tsv: tab-separated spreadsheet of the entire corpus (before preprocessing.) The first column denotes the movie review ID; the second column denotes the sentiment of the review (0: negative; 1: positive); the third column contains the text of the review. Each row represents a review of a film.
- 'counts.csv': document-term matrix extracted from the previous file after preprocessing (tokenizing, stemming…). The matrix has 24,998 rows (one per movie review) and 2073 terms (i.e., word stems). This is a very large matrix (over 51 million entries), and also very sparse (only 3.3% of the entries are non-zero). For this reason, the file 'counts.csv' only includes the row (movie review), column (term) and corresponding word count of non-zero cells. You will need to reconstruct the sparse matrix from it.
- 'vocabulary.csv': spreadsheet containing the word stems in the vocabulary.

- 'sentiment.csv': spreadsheet containing the sentiment (0: negative; 1: positive) for each of the movie reviews

The file 'reviews.tsv' is included for completeness, but you are not required to parse it since we are already providing the bag-of-words representation in the other three files.

a. **5% credit.** Use Latent Dirichlet Allocation (LDA) to generate a topic model with 10 topics. Display the top-10 words for each topic as a 10x20 matrix (each column being a topic) and guess what each of the 10 underlying topics may be.
b. **5% credit.** Remove the 100 most frequent words from the bag-of-words, regardless of sentiment. Then, repeat part a. Display the top-10 words for each topic and discuss what each of the 10 underlying topics may be, compared to those you obtained in part a.
c. **10% credit**. Find the top 500 movie reviews for each topic in part b[1], then compute the average sentiment of those 500 reviews. Does the average sentiment of each topic help you interpret those topics? Please elaborate.
d. **10% credit**. Split the corpus into two sets: X+ containing only positive reviews, and X- containing only negative reviews. Then compute the tf-idf matrix for each set: M+ from X+ (roughly a 12,500×2073 matrix), and M- from X- (~12,500×2073). Next, compute the average tf-idf of each word in X+ (i.e., a 1×2073 vector), and the average tf-idf of each word in X- (i.e., another 1×2073 vector). Finally, compute the absolute value of the difference between the two vectors, and sort words by decreasing order. Which words come at the top and at the bottom of the list? Can you interpret the result?

Submission guidelines

File naming convention

Please submit a separate ZIP file for each problem:

1. Problem 1: LastnameFirstname_p1.zip (e.g., GutierrezRicardo_p1.zip)
2. Problem 2: LastnameFirstname _p2.zip
3. …

Each ZIP file should include:

1. A PDF report (LastnameFirstname _p1.pdf)
2. A Jupyter notebook (LastnameFirstname _p1.ipynb)
3. Any data files your code needs in order to run (e.g., CSV files). No naming convention needed as long as it is consistent with your code.

PDF reports:

- Each PDF report will be graded in isolation from the associated code. Thus, please make sure the PDF includes all the necessary figures.
- Please use language that is appropriate for a technical report. Informal language (e.g., something you would write in an email to a friend) is not appropriate. Neither is the use of hyperbole. Technical terms should also be used rigorously (e.g., the term 'significant' is only appropriate when supported by an appropriate test statistic).
- Stuff you can do to lower your grade:

[1] For each movie review, LDA generates a score for each of the topics (the higher the score, the better that the movie review matches the corresponding topic).

    - Use unwarranted[2] precision on numerical results.
    - Use lots of text fonts, font sizes, colors, and ***lots*** of Emphasis Everywhere.
        - Change the paragraph alignment.
    - Refuse to use a spelchercker. (because that's for wimps)
    - Ignore the file naming conventions above (this one is a sure winner)
    - Include snippets of your code, especially with dark backgrounds (it's Halloween!)
    - Include screenshots of numerical results, especially when you can see the pixelation. For example, see Figure 1.
- Things you can do to earn 5% extra credit
    - Not do stuff that would lower your grade (see above)
    - Start each section of a problem on a new page[3].
    - Include a caption for each figure, describing its contents.
    - Discuss each figure and cross-reference it, just like we are doing with Figure 1.
    - Use **clearly identifiable** section headings (if unsure, refer to the syllabus)
    - Each figure has its axes labeled and includes a legend whenever possible
    - The legend is easily interpretable (e.g., {veggies, meat, …} instead of {class 1, class 2, …}
    - Use colors that make the figures easy to interpret
    - Each figure and its text are properly sized (i.e., the opposite of Figure 1.)

Figure 1. Look at my p!

Code:

- Please make sure your code runs before submitting it. This includes making sure the ZIP file has all the required data files.
- You will be allowed to make minor modifications (e.g., TypeError, NameError) to your code if it does not run initially. However, any modification to your submitted code will result in a 50% reduction on the code grade.
- Please add comments to your code to assist us in reviewing it.

You are free to use any code or notebooks that are available online, as long as you:

- Acknowledge the source,
- Provide a link to it (e.g., URL),
- Describe what the code does, and
- Describe how you modified it to serve your purpose

If you use existing code/notebooks without proper citations, your submission will be treated as a case of **plagiarism** (see Syllabus).

---

[2] Reporting a classification rate of 45.343543875514441% when you have 100 examples in the dataset is incorrect; the best you can do in that case is 1%.

[3] If you are concerned about saving trees, remember we're not printing the PDFs.