University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Data Mining
Lecturer: Dr. Alejandro Arbelaez

Declaration:

By submitting this assignment. I agree to the following:

"I have read and understand the UCC academic policy on plagiarism and I agree to the requirements set out thereby in relation to plagiarism and referencing. I confirm that I have referenced and acknowledged properly all sources used in preparation of this assignment.

I declare that this assignment is entirely my own work based on my personal study. I further declare that I have not engaged the services of another to either assist me in, or complete this assignment"

Submission:

This project is due on April/3/2020. You should submit a single file with your solution ([your-name-Project.zip] electronically via Canvas. This file must include your R code as well as a pdf file with your report.

Please note that this project will account for 50% of your module grade.

Objectives

This project is intended to give you hands-on experience in using a practical machine learning tool and analysing scientific results. You will first use your Naïve Bayes implementation of a previous lab and compare the performance of your algorithm against state-of-the-art supervised learning implementations in R.

In this work we will use an online newsgroup text dataset (available in Canvas newgroups.zip). This dataset contains text articles posted to each four online newsgroups,

for a total of 400 articles. For documentation and complete details please see the following website: http://www-2.cs.cmu.edu/afs/cs/project/theo-11/www/naive-bayes.html
This data is useful for a variety of text classification projects. The target label of each article is the newsgroup it bellows to.

Exploration of the dataset (10 Marks)

Your first task is to get a feel of the data that you will be dealing with in the rest of the assignment.

- Use tokenization and compute the top 200 most popular words (total occurrences)
- Repeat the previous step, but now filter tokens by length (min. size = 4 and max. size = 20). Please indicate the total occurrences of the words.

Basic evaluation: (15 Marks)

- Train and evaluate different classifiers using 70% of the dataset from training and 30% for testing. Try your Naïve Bayes implementation from a previous lab, knn and Random Forest (use frequency counts for knn and Random Forest). For this part of the project we are not interested in optimizing the parameters, we just want to get an idea of the dataset.

  Compare the results of the three classifiers.

Robust evaluation (25 Marks)

In this section we are interested in more rigorous techniques by implementing more sophisticated techniques.

- Hold-out and cross-validation.
- More classification algorithms, e.g., decision trees, and SVM.
- Hyper-parameter tuning.
- Feature selection and engineering.
- Performance metrics.
- Pre-processing techniques (with basic techniques, e.g., converting everything to lower-case, removal of punctuation, etc).
- etc.

Report (20 Marks)

The report should consists of two parts:

- The basic evaluation report should describe the results of your basic evaluation using tables, figures, and performance metrics (e.g., accuracy, confusion matrix, recall, etc.).

- The detailed and robust evaluation should describe the basic methods you have employed for cleaning the dataset (for example, converting everything to lower-case, removal of punctuation, etc). It should also provide an account of the performance of the model and how it was impacted by the basic methods of cleaning the data. Furthermore, you should clearly describe and report your results, with figures, tables, and performance metrics (to compare the performance of the ML models).

Code requirements: (30 Marks)

Program Correctness: Your program should work correctly on all inputs. Also if there is any specifications about how the program should be written, or how the output should appear, those specifications should be followed.

Readability: Variables functions should have meaningful names. Code should be organized into functions/methods where appropriate.
There should be an appropriate amount of white space so that the code is readable, and indentation should be consistent.

Documentation: your code and functions/methods should be appropriately commented. However, not every line should be commented because that makes your code overly busy. Think carefully about where comments are added.

Code Elegance: There are many ways to write the same functionality into your code, and some of them are needlessly slow or complicated. For example, if you are repeating the same code, it should be inside creating a new method/function or for loop

Code efficiency: The implementation is logically well designed without inappropriate design choices (e.g., unnecessary loops).