Assignment
Designing a Spam Filter
B.Sc Data Science II

D. T. McGuiness, PhD

version WS 2024

Topic	Description
Module	Data Science II
Module Code	MLDS
Semester	5
Lecturer	D. T. McGuiness, PhD
ECTS	5
SWS Total	2
SWS Lab	0
SWS Teaching	2
Lecture Type	ILV
Teaching UE	28 + 2
Coursework Name	Assignment - Designing a Spam Filter
Work	Individual
Suggested private Study	20 hours including Report Writing
Submission Format	Online via SAKAI
Submission Deadline	5th January 23:59
Late Submission	Not accepted
Resubmitting Opportunity	No re submission opportunity
Feedback	Via Comments on SAKAI
Learning Outcomes	- Understanding of Data Filtering
	- Understanding of Classification Methods
	- Understanding of ML Pipelines
	- Understanding of Python Programming

Table 1: Information about the assignment.

> No lecture time is exclusively devoted to the aforementioned assignment.

A portion of the mark for every assignment will be, where applicable, based on style. Style, in this context, refers to organisation, flow, sentence and paragraph structure, typographical accuracy, grammar, spelling, clarity of expression and use of correct IEEE style for citations and references. Students will find *The Elements of Style (3rd ed.)* (1979) by Strunk & White, published by Macmillan, useful with an alternative recommendation being *Economist Style Guide (12th ed.)* by Ann Wroe.

Assignment

The assignment is focused on building a **spam classifier**.

Spam Filter

A program used to detect unsolicited, unwanted and virus-infected emails and prevent those messages from getting to a user's inbox. Like other types of filtering programs, a spam filter looks for specific criteria on which to base its judgments.

The steps you should take into finishing this assignment is as follows:

a. Download examples of spam and ham from Apache Spam Assassin's public datasets.

b. Unzip the dataset and familiarise yourself with the data format and apply the necessary data processing.

c. Split the data into training set and test set.

d. Write a data preparation pipeline to convert each email into a feature vector. Your preparation pipeline should transform an email into a (sparse) vector that indicates the presence or absence of each possible word.

For example, if all emails only ever contain four (4) words:

"Hello", "how", "are", "you",

then the email "Hello you Hello Hello you" would be converted into a vector:

$$[1, 0, 0, 1]$$

Which means:

["Hello" is **present**, "how" is **absent**, "are" is **absent**, "you" is **present**],

> Or $[3, 0, 0, 2]$ if you prefer to count the number of occurrences of each word.

You may want to add hyperparameters to your preparation pipeline to control whether or not to strip off email headers, convert each email to lowercase, remove punctuation, replace all URLs with "URL", replace all numbers with "NUMBER", or even perform stemming.

> such as trimming off word endings which Python has libraries available to do this.

e. Finally, try out several classifiers and build a spam classifier with **both high recall and high precision**.

What is Required of You

For this work you are required to submit a report. Due to its nature of significant programming you can submit your report as an iPython notebook.

However you can also submit a report in LaTeX as well.

In your report make sure you explain your code using snippets in your report and also submit the code as a python script as well.

The final grade of this assignment will be based on:

1. Documentation of your code.
2. Final recall and high precision.
3. Explanation of the decisions taken.

Style Guidelines

Formatting

1. Your report should be organized and in order.
2. Code should be single spaced and use readable font sizes.
3. Use proper code indentation. If you paste code into LaTeX or Word, make sure that the indentation still looks correct.
4. Add spacing around operators if it improves clarity.

> To have a good overview of industrial standards of python programming, please have a look at PEP 8 - Style Guide for Python Code.

Commenting

1. The goal of commenting is to make your code easily understandable by someone else.
2. There should be a block comment at the start of your program explaining what it does and including: your name, the assignment and the date.
3. Every method should have a block comment explaining what is does. I suggest that you write the block comment before you write the method.
4. Add inline comments where needed to explain subtle or tricky code.
5. Add blank lines and comments to separate sections of large methods that perform several different tasks.

Naming Variables, Methods, and Classes

1. Take the time to choose meaningful names.
2. Follow naming conventions (i.e., PEP).
3. You can use abbreviations, but add comments explaining them.

> Many people comment out old code that wasn't working correctly. Clean up and remove any old code before turning it in.