Implementing your own spam filter

Implementing your own spam filter

While doing this hands-on exercise, you'll work with *natural language data*, learn how to detect the words spammers use (https://sm.asisonline.org/migration/words-spammers-use-lure-you-

006874.aspx) automatically, and learn how to use a *Naive Bayes classifier* for binary classification.

Classification

Classification is ubiquitous: many things around us can be divided into two or more classes based on their characteristics. For example, we call a vehicle with one wheel a “unicycle”, a vehicle with two wheels and no motor a “bicycle”, and a vehicle with two wheels and a motor a “motorcycle”. Here, we

are using the *number of wheels* and the *presence of a motor* to classify the vehicle. Machines can do the same. Even better, being exposed to a sufficient amount of data describing the instances of different classes, they can learn about their characteristic properties and detect the classes of the new instances.

A special case of classification is *binary classification* which assumes that the choice is

between just two classes.

Spam filtering

Spam filtering is a binary classification task familiar to any user of email services. Now you'll learn how to implement your own spam filter.

The task is to distinguish between two types of emails, "spam" and "non-spam" often called "ham". The machine learning classifier will detect that an email is spam if it is characterised by certain features.

The textual content of the email – words like “Viagra” or “lottery” or phrases like “You've won a 100,000,000 dollars! Click here!”, “Join now!” – is crucial in spam detection and offers some of the strongest cues:

Since you’ll be working with text data, you’ll use the Python-based library Natural Language Toolkit (http://www.nltk.org) (NLTK). It is well supported and well

documented. You can even read the online book (http://www.nltk.org/book/).

Start by importing the toolkit:

```
import nltk
```

To train the classifier, we need a representative dataset with both spam and ham emails. We'll be using the Enron email dataset (http://labs-repos.iit.demokritos.gr/skel/i-config/downloads/enron-

spam/preprocessed/) which contains emails of both types stored in plain text format.

You can download the archive and extract any of the directories – for example, the `enron1/` directory contains 3672 legitimate (ham) emails and 1500 spam emails.

The spam detection algorithm will involve five steps:

- Lodaing the data,
- Preprocessing,
- Extracting the features,

- Training the classifier, and
- Evaluating the classifier.

Let's get started!

Step 1: Loading the data

You need to read in the files from the spam and ham subfolders and keep them in two separate lists. To be able to iteratively read the files in a folder, add the following import statement:

```
import os
```

Then define the function `initialise_lists` as follows: