

1. [100 points] Spam classification

In this problem, we will use the Naive Bayes algorithm and an SVM to build a spam classifier.

In recent years, spam on electronic media has been a growing concern. Here, we'll build a classifier to distinguish between real messages, and spam messages. For this class, we will be building a classifier to detect SMS spam messages. We will be using an SMS spam dataset developed by Tiago A. Almedia and José María Gómez Hidalgo which is publicly available on <http://www.dt.fee.unicamp.br/~tiago/smsspamcollection>¹

We have split this dataset into training and testing sets and have included them in this assignment as `src/spam/spam_train.tsv` and `src/spam/spam_test.tsv`. See `src/spam/spam_readme.txt` for more details about this dataset. Please refrain from redistributing these dataset files. The goal of this assignment is to build a classifier from scratch that can tell the difference the spam and non-spam messages using the text of the SMS message.

- (a) **[25 point(s) Coding]** Implement code for processing the the spam messages into numpy arrays that can be fed into machine learning models. Do this by completing the `get_words`, `create_dictionary`, and `transform_text` functions within our provided `src/spam/spam.py`. Do note the corresponding comments for each function for instructions on what specific processing is required.

The provided code will then run your functions and save the resulting dictionary into `spam_dictionary` and a sample of the resulting training matrix into `spam_sample_train_matrix`.

- (b) **[25 point(s) Coding]** In this question you are going to implement a Naive Bayes classifier for spam classification.

Code your implementation by completing the `fit_naive_bayes_model` and `predict_from_naive_bayes_model` functions in `src/spam/spam.py`.

Now `src/spam/spam.py` should be able to train a Naive Bayes model, compute your prediction accuracy and then save your resulting predictions to `spam_naive_bayes_predictions`.

Remark. (OPTIONAL!) If you implement Naive Bayes the straightforward way, you will find that the computed $p(x|y) = \prod_i p(x_i|y)$ often equals zero. This is because $p(x|y)$, which is the product of many numbers less than one, is a very small number. The standard computer representation of real numbers cannot handle numbers that are too small, and instead rounds them off to zero. (This is called “underflow.”) You'll have to find a way to compute Naive Bayes' predicted class labels without explicitly representing very small numbers such as $p(x|y)$. [**Hint:** Think about using logarithms.]

- (c) **[25 point(s) Coding]** Intuitively, some tokens may be particularly indicative of an SMS being in a particular class. We can try to get an informal sense of how indicative token i is for the SPAM class by looking at:

$$\log \frac{p(x_j = i|y = 1)}{p(x_j = i|y = 0)} = \log \left(\frac{P(\text{token } i | \text{email is SPAM})}{P(\text{token } i | \text{email is NOTSPAM})} \right).$$

¹Almeida, T.A., Gómez Hidalgo, J.M., Yamakami, A. Contributions to the Study of SMS Spam Filtering: New Collection and Results. Proceedings of the 2011 ACM Symposium on Document Engineering (DOCENG'11), Mountain View, CA, USA, 2011.

Complete the `get_top_five_naive_bayes_words` function within the provided code using the above formula in order to obtain the 5 most indicative tokens.

- (d) **[25 point(s) Coding]** Support vector machines (SVMs) are an alternative machine learning model that we discussed in class. We have provided you an SVM implementation (using a radial basis function (RBF) kernel) within `src/spam/svm.py` (You should not need to modify that code).

One important part of training an SVM parameterized by an RBF kernel (a.k.a Gaussian kernel) is choosing an appropriate kernel radius parameter.

Complete the `compute_best_svm_radius` by writing code to compute the best SVM radius which maximizes accuracy on the validation dataset.