The University of Melbourne
School of Computing and Information Systems
COMP30027 Machine Learning, 2025 Semester 1

Project 1: Scam detection with naïve Bayes

Due:	7 pm, 11 April 2024
Submission:	Source code (in Python) and written responses
Marks:	The project will be marked out of 20 points and contribute 20% of your total mark.

Overview

In this project, you will implement supervised and semi-supervised naïve Bayes models to detect scam SMS messages using text features. You will perform some analyses on a provided dataset and use these to answer conceptual questions. You will then have a choice of how to extend your model for further experiments and evaluation.

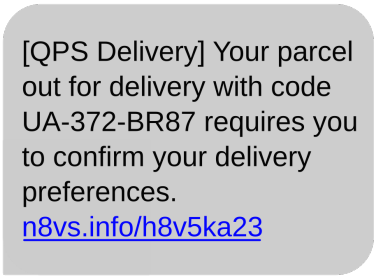

Figure 1: Example of a phishing SMS message

Problem description and dataset

Phishing is a type of cyberattack in which the attacker tries to trick a victim into revealing sensitive information, such as passwords or bank details, or install malware. An example of an SMS phishing attack (a.k.a "smishing") is shown in Figure 1. In this example, an attacker sends a message which appears to be from a delivery company, in order to trick the recipient into visiting a malicious website.

In this assignment, you will develop a machine learning model to classify messages as either "scam" or "non-malicious." The data for this assignment is derived from the SMS PHISHING DATASET [1], a large collection of real-world SMS messages. Messages have been labelled as either "scam" (a label which includes both phishing and spam) or "non-malicious." The dataset is provided in .csv file format with the following columns:

- `textOriginal`: the original message text
- `textPreprocessed`: pre-processed text for use in machine learning models (the processing removes very common words like "the," removes very rare words, and converts words to a standardised format)
- `class`: a boolean class label (0=non-malicious, 1=scam)

In this assignment, you should use `textPreprocessed` as the attributes (input) to your prediction model, and attempt to predict the boolean label `class`.

We have split the dataset into 3 partitions: a supervised training set `sms_supervised_train.csv`, an unlabelled training set `sms_unlabelled.csv`, and a test set `sms_test.csv`. Please use the provided dataset split throughout this assignment.

Please note that the dataset is a sample of real-world SMS messages. As such, it may contain information that is in poor taste, or that could be construed as offensive. We would ask you, as much as possible, to look beyond this to the task at hand. (For example, it is not actually necessary to read the original SMS messages `textOriginal`; we have only included them for transparency in case you are curious about the text pre-processing steps.)

Multinomial naïve Bayes implementation

To classify text, you will implement a multinomial naïve Bayes classifier. This model predicts a class label based on the number of times individual words appear in a message. This model treats each word in a message as an independent sample from a probability distribution of possible words. This is not really a correct model for natural language – words in a message are not independent and word order matters. However, this simple model can work well for many classification tasks.

The steps to train the model are as follows:

1. Define the **vocabulary** for this task, which is a list of every word which occurs in the training dataset (every word in `textPreprocessed`). Note that although we'll refer to the pre-processed tokens as "words" throughout this assignment, they also include non-word tokens like punctuation marks and digits; you can treat these as "words" for purposes of machine learning.

2. Define the **count** matrix. This is a matrix of size $N \times V$ where N is the number of instances in the training data and V is the number of words in the vocabulary. Each cell in the matrix represents the number of times a given word appeared in a given message (note that this matrix will be sparse – most of the values should be 0).

3. Compute the prior probability of each class $P(c)$. As we showed in lecture, this can be computed as:

$$P(c) = \frac{N_c}{N} \tag{1}$$

   where N_c is the number of instances in class c and N is the number of instances in the training data.

4. For each class c, for each word i, calculate the probability $p_{c,i}$, the probability of word i appearing in messages from class c. This can be computed without smoothing as:

$$p_{c,i} = \frac{count_{c,i}}{total_c} \tag{2}$$

   or with Laplace smoothing as:

$$p_{c,i} = \frac{count_{c,i} + \alpha}{total_c + V\alpha} \tag{3}$$

where $count_{c,i}$ is the total count of times word i appears in messages from class c, $total_c$ is the total count of words in class c, and α is the smoothing parameter. Note that the total probability within each class should sum to 1!

$$\sum_i p_{c,i} = 1 \tag{4}$$

To classify a test instance:

1. Compute a count vector for the test instance. This is a vector of length V which represents the number of times each word in the training set vocabulary appears in the test instance.
2. For each class, compute the posterior probability of the class conditional on the observed word count.

   $$P(c|count) \propto P(c)P(count|c) \tag{5}$$

   Use the multinomial distribution to compute the likelihood of the test item's word count conditional on class (x_i is the count of word i in the test instance):

   $$P(count|c) = \frac{n!}{x_1!x_2!...x_V!} p_{c,i}^{x_1} p_{c,1}^{x_1} p_{c,2}^{x_2} ... p_{c,V}^{x_V} \tag{6}$$

You may use built-in functions to implement any/all of these steps, or write your own functions.

Note that is possible for test instances to include words that aren't in your trained model's vocabulary (**out-of-vocabulary** words); these words should simply be ignored when counting. If a test instance contains NO words from the learned vocabulary, it should be skipped entirely, since it will not be possible to predict a test label for this instance.

1. Supervised model training [3 marks]

Use the provided supervised training dataset `sms_supervised_train.csv` to train a multinomial naïve Bayes model as described above. In your write-up, report and briefly discuss what this model has learned. Your write-up should answer the following questions:

1. What are the prior probabilities of the classes $P(c)$ in the training dataset?
2. What are the most probable words in each class? List about 10 per class and their probability values $p_{c,i}$.
3. What individual words are most strongly predictive of each class? One way to evaluate this is to calculate the probability ratio for a word occurring in class $c1$ versus class $c2$:

   $$R = \frac{p_{c_1,i}}{p_{c_2,i}} \tag{7}$$

   Higher values of R indicate that the word is more likely to appear in class c_1 than class c_2.

   What 10 words are most strongly predictive of the **scam** class? What 10 words are most strongly predictive **non-malicious** class? List the words and the probability ratios R.

In addition to reporting these points, briefly discuss them. Do these result seem reasonable? Do you think it will be possible to distinguish the two classes using the multinomial naïve Bayes model? Why? Your response should be no more than 250 words.

2. Supervised model evaluation [5 marks]

Use your trained classifier to predict the labels of the test dataset `sms_test.csv`. In your write-up, report and discuss what your model has learned. Your write-up should address the following points:

1. Report the overall accuracy of your classifier as well as a breakdown by class (e.g., using a confusion matrix or by reporting precision and recall).
2. Report how often your model encountered out-of-vocabulary words in the test set. Were any test items skipped (not classified) due to this problem?
3. Provide examples of instances classified with high and low confidence. A naïve Bayes model returns a posterior likelihood for each class $P(c_i|instance)$, so we could take the ratio of these as a measure of confidence:

$$R = \frac{P(c_1|instance)}{P(c_2|instance)} \tag{8}$$

   Higher values of R means the model is more confident that the class is c_1, while $R = 1$ means the model considers both classes equally likely. Provide some examples of test instances (at least 3 each, along with the R values) which are

   (a) classified **scam** with high confidence

   (b) classified **non-malicious** with high confidence

   (c) on the boundary between the two classes (R near 1)

In addition to reporting these points, briefly interpret them. Do the model's high and low confidence predictions seem reasonable to you? Is one class easier to identify than the other? Your response should be no more than 250 words.

3. Extending the model with semi-supervised training [8 marks]

In addition to the supervised training data, we have provided a separate unlabelled[1] dataset in the file `sms_unlabelled.csv`. You can use this unlabelled data to extend your model's capabilities through semi-supervised learning. Semi-supervised models learn from a combination of labelled and unlabelled training data.

Choose **ONE (1)** of the following approaches to extend your model:

Option 1: Label propagation

In this approach, you will use a supervised model to apply labels to the unlabelled data. First, use your model from Q1 to classify the unsupervised dataset. Treat these model predictions as the "ground

[1] We have provided the ground truth labels for the "unlabelled" data just to support option 2 (active learning). **You should not use the provided class labels when doing unsupervised training.** You also **should not** use these labels as part of your model selection or parameter tuning. You may refer to the provided labels to help you debug your code, or as part of your evaluation for your write-up.

truth" labels. Then retrain your model with an expanded dataset consisting of the supervised dataset from Q1 plus the unsupervised data with model-generated "ground truth" labels. Consider these questions to improve on the basic label propagation approach:

- Is it better to use all of the unlabelled data as described above, or only the instances that the Q1 model could label with high confidence?
- Do you get a different result if you iteratively update the model, retraining on only a portion of the unlabelled data each time? For example, use the Q1 model to label 50% of the unlabelled data, retrain, use the retrained model to classify the remaining 50% of the data, and then train the final model?

Option 2: Active learning

In this approach, you will select a small portion of the "unlabelled" data (200 instances), reveal the labels, and use these instances for supervised learning. One simple way to do this is to select 200 instances at random. Retrain your model from Q1 with an expanded dataset consisting of the supervised dataset from Q1 plus the additional 200 instances you selected. (The other unlabelled instances should be ignored.) Consider these questions to improve on the basic active learning approach:

- Some of the unlabelled instances are probably more informative than others. Is there a way you could select 200 instances that provide the most information, or provide information that is not already available in the supervised training data?
- Can you use the Q1 model to help guide the selection? (For example, select instances where the model's prediction is high or low confidence?)

For purposes of this assignment, pick **only ONE (1)** of the two approaches above and implement it with your Q1 model and the provided unsupervised dataset. Experiment with parameters to try to find an implementation that works well. You should use a validation set to choose parameters – don't just select the parameters that maximize accuracy on the test set.

We have provided ground truth labels for the "unlabelled" data, but you should ignore these when doing unsupervised training. If you choose option 1 (label propagation), you should not use the provided labels at all in your method. If you choose option 2 (active learning), you may only use the labels from the 200 instances you selected. **Do not** use the other labels to guide any aspect of your model design (for example, do not use the full set of labels to decide which 200 instances to select).

In your write-up, explain your approach and give the reasoning behind your implementation choices. Show results on the training and/or validation set to support your discussion. Your response should be no more than 500 words.

4. Supervised model evaluation [4 marks]

Critically evaluate your Q3 model on the test dataset. Your write-up should answer the following questions:

1. How does the performance of your semi-supervised model (Q3) compare to your supervised model (Q1)?
2. How has the model's representation changed after the semi-supervised training? To answer this question, you might investigate whether there are any changes in the model's confidence or any changes in which words the model considers most predictive of each class.

Your response should be no more than 500 words.

Implementation tips

You're welcome to use any library functions you wish to complete this assignment. If you use library functions, please check the documentation and make sure you understand how to access the model parameters and results needed to answer the questions above.

If you write your own functions, bear in mind that you may run into underflow problems when multiplying very small probabilities. Instead of taking the product of probabilities p_i

$$\prod_i p_i \tag{9}$$

it may be safer to compute the sum of the log of those probabilities

$$\sum_i log(p_i) \tag{10}$$

Submission

Submission will be made via the LMS. Please submit your code and written report separately:

- Your code submission should use the provided .ipynb notebook template. Your submission must include comments or a README section that explain how to run your code so we can reproduce your results.
- Your written report should be uploaded separately as a .pdf, using the Turnitin submission link. Please keep your responses within 250 words (each) for Q1 and Q2, and within 500 words (each) for Q3 and Q4. Please ensure that all of the results, tables, graphs, figures, etc. which you wish to include in your answer are included in your written report. Results and figures which appear only in the Jupyter notebook but not in the report will not be marked.

Late submission

The submission mechanism will stay open for one week after the submission deadline. Late submissions will be penalised at 10% per 24-hour period after the original deadline.

To request an extension on this assignment, please see the FEIT extension policy and follow the steps below:

- **To request an extension of 1-3 business days (without AAP)**, complete the FEIT Extension Declaration Form at the website above and upload it to Canvas under **Assignment 1 extension request**. We do not accept these forms by email in COMP30027; the form *must* be uploaded to Canvas.
- **To request a longer extension (without AAP)**, please apply for Special Consideration.
- **If you have an AAP**, please request an extension by completing the **Assignment 1 extension request** form on Canvas and uploading your AAP.

Please note that we can only accept extension requests via Canvas up until the assignment deadline. Late extension requests can only be granted through Special Consideration. The longest extension granted on this assignment is 10 business days.

Assessment

Marks will be alloted to the individual sections of this assignment as indicated above. When marking your report, we will be looking for evidence that you have a correct implementation that allows you to explore the problem, but also that you have thought deeply about the data and the behaviour of your models. Code submissions will only be assessed for correctness (does the code actually implement the intended machine learning method, as described in this spec / your report).

Updates to the assignment specifications

If any changes or clarifications are made to the project specification, these will be posted on the LMS.

Academic misconduct

You are welcome — indeed encouraged — to collaborate with your peers in terms of the conceptualisation and framing of the problem. For example, we encourage you to discuss what the assignment specification is asking you to do, or what you would need to implement to be able to respond to a question.

However, sharing materials — for example, showing other students your code or colluding in writing responses to questions — or plagiarising existing code or material (including materials generated by ChatGPT or other AI) will be considered cheating. Your submission must be your own original, individual work. We will invoke University's Academic Misconduct policy where inappropriate levels of plagiarism or collusion are deemed to have taken place.

References

[1] Mishra, S. & Soni, D. SMS PHISHING DATASET FOR MACHINE LEARNING AND PATTERN RECOGNITION. Mendeley Data, V1, 2022. doi: 10.17632/f45bkkt8pr.1