CSEC 520/620: Cyber Analytics & Machine Learning

Assignment 1 - SMS Spam Detector

Due Date - September 15, 2022 11:59 PM

Purpose

The purpose of this assignment is to learn about two basic machine learning techniques and how they can be used to solve one real world cyber analytics problem.

Description

You and your team will develop two SMS spam classifiers using:

- *k* Nearest Neighbors (*k*-NN), and
- Multinomial Naive Bayes (usually just "Naive Bayes").

You are NOT allowed to use any third-party libraries (e.g. scikit-learn, etc.) to implement the classifiers.

Dataset

Get the dataset and readme file from MyCourses. Read the readme file to learn about the data and familiarize yourself with the dataset.

Steps

1. **Tokenizing**
   a. You will need to write code to read in the dataset file and to 'tokenize' the messages. The standard tokenization strategy is to split each message into its space-separated words. You can also examine 'cleaning' strategies to improve the quality and meaning of your tokens.
   b. After the data has been loaded and tokenized, the dataset must be split into train and test subsets before building and evaluating the following classifiers.

2. ***k* Nearest Neighbors**
   In this section, you will learn how to use *k*-NN for spam identification.
   a. To begin, each SMS sample will need to be converted to a numerical representation before the neighbor distances can be calculated. For this, you will convert each SMS to a **TF-IDF** feature vector.
      i. Identify the list of unique terms in the train set (this will determine your feature vector).
      ii. Calculate the inverse document frequency (*IDF*) for each term using your train set.
      iii. For each sample in both train and test sets, calculate the term frequency (*TF*) for each term in the sample, then use the IDF values to calculate the TF-IDF for terms identified in step (a). This TF-IDF vector is the sample's feature vector.
      iv. Refer to http://www.tfidf.com/ for additional details.

b. Numerical vectors can be classified using k-NN by comparing the distance between a test sample and all training samples. To identify the predicted label, use the most commonly appearing label in the k number of closest training samples.

3. **Naive Bayes Classifier**

   In this section of the assignment, you will develop a Multinomial Naive Bayes classifier.

   a. A Naive Bayes classifier works by predicting the probability that a message is spam or not spam given the collection of words used within the message. The probability of spam given a SMS can be represented by the following formula:

   $$P(Spam|W) = P(Spam) * \prod_{i=1}^{n} P(w_i|Spam)$$

   where $\prod_{i=1}^{n} P(w_i\,|Spam)$ is the product of $P(w|Spam)$ for every word (w) in the collection of words representing the sample (W), and $P(Spam)$ is the percentage of spam SMS in the dataset.

   b. The probability of each word given spam or not spam must be calculated before $P(Spam|W)$ can be computed. $P(w|Spam)$ can be calculated by dividing the frequency of the word w in the *Spam* subset by the total number of words in the *Spam* subset.

   c. To predict the label of an unknown sample, compute both $P(Spam|W)$ and $P(not\ Spam|W)$. The higher value is the label predicted by the Naive Bayes classifier.

4. **Performance metrics**

   To analyze the performances of your implemented k-NN and Naive Bayes classifiers, you will need to include the following metrics in your code:

   a. Correct/incorrect classification:
      i. True positive, TP = Correct classification of spam SMS
      ii. False positive, FP = Incorrect classification of non-spam SMS as spam,
      iii. True negative, TN = Correct classification of non-spam SMS
      iv. False negative, FN = Incorrect classification of spam SMS as non-spam

   b. Accuracy = (TP + TN) / (TP + FP + TN + FN)

   c. Precision = TP / (TP + FP)

   d. Recall = TP / (TP + FN)

   e. F1-score = (2 * Precision * Recall) / (Precision + Recall)

Deliverables

1. All source code of your implementations.
   - Document your code! Your code documentation should demonstrate you understand what your code is doing (and that you did not just copy-and-paste from an external source).
2. A readme file that should contain:
   - A clear description of the directions to set up and run your code.
   - If your code requires external libraries, or if not written in Python, provide the additional references/direction.

   If your instructions are not clear, your work will ***NOT*** *be graded!*
3. A short report describing your experiments and the results of your evaluations.
4. **Alternatively**: If using GoogleColab & JupyterNotebook you can submit source code, readme, and report as a single .ipynb file.
   - DO use the table of contents and sections to allow for easy navigation.
   - DO NOT intersperse results/analysis/discussion between code-blocks. The *Expected Report Elements* should be easy to identify and read separate from the code execution.

Expected Report Elements:

1. **Include the following results. Use figures/tables/graphs/etc. as appropriate**:
   a. Compare the accuracy of the train and test dataset for both the classifiers. Discuss if your classifier has any overfit/underfit issue.
   b. Provide the precision, recall, accuracy, and F1-score values of the k-NN and Naive Bayes classifiers.

2. **Analysis. Consider and briefly discuss the following questions**:
   a. How does the choice of k and/or distance metrics affect the performance of the nearest neighbors algorithm? Include graphs to illustrate your findings.
   b. You have learned from the video and book that there is no training time in k-NN. Report your observation about this from your implemented k-NN.
   c. How does the performance and efficiency of the two algorithms compare? Is one better than the other? Why do you think that may be the case?
   d. Are there techniques that can be used to improve the performance of these classifiers (e.g. improvements to text tokenization or the Naive Bayes/k-NN algorithms)?
      i. **Bonus**: Compare the results of applying these techniques with the original classifiers.
   e. What are the shortcomings and limitations of each classifier?

3. **Provide answers to the following exercises**:
   a. Suppose that the Accuracy of a classifier on this dataset was 94.8%, with differing rates of false positives and false negatives (FNR = 10% and FPR = 4%). If 10,000 samples are analyzed and spam messages appear at a base-rate of 20%, compute:
      i. The expected number of TPs, TNs, FPs, and FNs.
      ii. The Precision and Recall values.
      iii. Suppose that you are an intern in the RIT ITS spam division, and you have been tasked with manually checking all the messages detected as SMS spam using this classifier to see if they are spam or not. It takes you one hour to check 100 messages. How much of your time will be spent looking at actual spam? How much will be spent looking at non-spam?
   b. Provide the detailed calculations and results in your report for the following toy example. You are preparing a simple Bayes classifier using the following word frequency dictionaries:

      W_{Spam} = [“click”: 2, “dude”: 1, “prize”: 3, “for”: 3, “look”: 1, “winner”: 3]
      $W_{not\ Spam}$ = [“babe”: 1, “dude”: 3, “look”: 2]

      i. Calculate $P(w|Spam)$ and $P(w|not\ Spam)$ for every word $W_{Spam}$and $W_{not\ Spam}$.
      ii. Using your results from *i.* compute $P(Spam|W)$ and $P(not\ Spam|W)$ for the following SMS when assuming that messages are spam 10% of the time. The SMS is *“dude! dude! look!”*.
         1. Is the message spam or not spam?
      iii. Do the same calculations for the known spam: *“winner babe! click for prize”*.
         1. What does the classifier say about this message? Why?
         2. Consider how this issue may be solved when building your classifier.

4. **Include any citations in an appropriate and consistent format**.

5. ***Not Expected*:** Background on the classifiers or SMS or SMS spam. Especially if it leads you to copy other material (see “A Note on Plagiarism” below!).

Grading Rubric

Criteria	1 Poor	2 Basic	3 Proficient	4 Distinguished
Model Implementation (25%)	No or non-functioning code for classifiers.	Classifiers are functional, but one (or both) implementations have significant errors.	Both classifiers are correct and well-functioning, however code quality is lacking and/or classifiers may produce minor errors.	Classifiers are correct and well described. Code is very cleanly written and is capable of handling most edge cases.
Experimental Results (25%)	Minimal results are reported.	Results are incomplete and/or wrong, or are not presented in a way where its meaning is clear.	A reasonable set of results, showing expected performance in most tests. Some presentation issues, such as confusing graphs/tables or unnecessary detail.	Classifiers achieve expected performance and are presented clearly.
Results Analysis (30%)	No analysis, or nonsensical analysis, provided. Missing most questions.	Analysis is inaccurate or hard to understand. Some, but not all, questions are addressed.	Analysis is sensible and covers most questions well. May be missing appropriate references or are weak in some areas.	Clear and accurate analysis that is backed up with appropriate references.
Writing Quality (10%)	Very poorly written and hard to follow.	Major points are visible, but writing may include many errors and/or lack focus and is disorganized.	Writing is clear enough to be understood, but some points may lack focus. Relatively few writing errors.	The paper is clear and well organized. Writing is smooth and polished with very few errors.
Code Documentation (10%)	Inadequate documentation provided both in the code and with the code.	Acceptable documentation for running the code, but lacking in-code documentation and helpful descriptions to demonstrate understanding.	Code includes some documentation (e.g. function docstrings, inline comments), however quality may be weak or unclear.	High quality documentation is provided. The purpose and function of all segments of code can easily be understood.

A: 3.25 average or higher
B: 2.5-3.25 average
C: 1.75-2.5 average

A Note on Plagiarism

When writing, you must include in-line citations whenever you are reporting on information that is not "general knowledge," i.e. anything you learned for this project and didn't know in advance. This is **NOT** just for quoted information. Failure to do this is plagiarism.

This article on plagiarism is good and covers the line between common knowledge and other material: https://writingcenter.unc.edu/tips-and-tools/plagiarism/. Also: https://www.plagiarism.org/ has a ton of additional information.

I have had students fail to follow these guidelines and get caught nearly every time I've taught my research seminar. These students get put on probation and have even been suspended from the university for this serious academic violation. *Please do not be the next!*