Student Name <- replace with your name

CS 481 Spring 2023 Programming Assignment #02

Due: **Sunday, April 2, 2023 at 11:59 PM CST**
Points: **100**

Instructions:

1. Place **all your deliverables (as described below) into a single ZIP** file named:

```
LastName_FirstName_CS481_Programming02.zip
```

2. Submit it to Blackboard Assignments section before the due date **[presentation slides can be added AFTER you presented]**. **No late submissions will be accepted**.

Objectives:

1. (100 points) Implement and evaluate a Naïve Bayes classifier algorithm.

Task:

Your task is to implement, train, and test a Naïve Bayes classifier using a publicly available data set.

Data set:

Pick a publicly available data (**follow the guidelines provided in Blackboard**) set first and do an initial exploratory data analysis. Note

Deliverables:

Your submission should include:

- Python code file(s). Your py file should be named:

  ```
  cs481_P02_AXXXXXXXX.py
  ```

  where AXXXXXXXX is your IIT A number (this is REQUIRED!). If your solution uses multiple files, makes sure that the main (the one that will be run to solve the problem) is named that way and others include your IIT A number in their names as well.

- Presentation slides in PPTX or PDF format. **Slides can be added to your submission AFTER you presented your work in class [You can resubmit everything then].** Name it:

  ```
  LastName_FirstName_CS481_P02_Slides.pptx or pdf
  ```

- This document with your observations and conclusions. You should rename it to:

`LastName_FirstName_CS481_P02.doc or pdf`

Implementation:

Your task is to implement, train, and test a Naïve Bayes classifier (as outlined in class) and apply it to classify sentences entered using keyboard.

Your program should:

- Accept one (1) command line argument, i.e. so your code could be executed with

  ```
  python cs481_P02_AXXXXXXXX.py IGNORE
  ```

  where:

  - `cs481_P02_AXXXXXXXX.py` is your python code file name,
  - `IGNORE` is a `YES` / `NO` switch deciding if your implementation will skip one pre-processing step (the one selected by you in Google Spreadsheet for the assignment),

  Example:

  ```
  python cs480_P01_A11111111.py YES
  ```

  If the number of arguments provided is NOT one (none, two or more) the argument is neither `YES` nor `NO` assume that the value for `IGNORE` is `NO`.

- Load and process input data set:
  - Apply any data clean-up / wrangling you consider necessary first (mention and discuss your choices in the Conclusions section below).
  - Text pre-processing:
    - treat every document in the data set as a single sentence, even if it is made of many (no segmentation needed),
    - if `IGNORE` is set to `YES`, skip one (selected earlier) of the steps below:
      - apply lowercasing,
      - remove all stop words (use the `stopwords` corpora for that purpose),
      - stem your data using NLTK's Porter Stemmer (NO lemmatization necessary)

- Train your classifier on your data set:
  - assume that vocabulary V is the set of ALL words in the data set (after pre-processing above),
  - divide your data set into:
    - training set: FIRST (as appearing in the data set) 80% of samples / documents,
    - test set: REMAINING 20 % of samples / documents,

   - use **binary** BAG OF WORDS with **"add-1" smoothing** representation for documents,
   - train your classifier (find its parameters. HINT: use Python dictionary to store them),

- Test your classifier:
   - use the test set to test your classifier,
   - calculate (and display on screen) following metrics:
      - number of true positives,
      - number of true negatives,
      - number of false positives,
      - number of false negatives,
      - sensitivity (recall),
      - specificity,
      - precision,
      - negative predictive value,
      - accuracy,
      - F-score,

- Ask the user for keyboard input (a single sentence S):
   - use your Naïve Bayes classifier to decide (HINT: use log-space calculations to avoid underflow) which class S belongs to,
   - display classifier decision along with P(CLASS_A |S) and P(CLASS_B | S) values on screen

Your program output should look like this (if pre-processing step is NOT ignored, output `NONE`):

```
Last Name, First Name, AXXXXXXXX solution:
Ignored pre-processing step: STEMMING

Training classifier…
Testing classifier…
Test results / metrics:

Number of true positives: xxxx
Number of true negatives: xxxx
Number of false positives: xxxx
Number of false negatives: xxxx
Sensitivity (recall): xxxx
Specificity: xxxx
Precision: xxxx
Negative predictive value: xxxx
Accuracy: xxxx
F-score: xxxx

Enter your sentence:
```

```
Sentence S:

<entered sentence here>

was classified as <CLASS_LABEL here>.
P(<CLASS_A> | S) = xxxx
P(<CLASS_B> | S) = xxxx

Do you want to enter another sentence [Y/N]?
```

If user responds `Y`, classify new sentence (you should not be re-training your classifier).

where:

- `xxxx` is an actual numerical result,
- `<entered sentence here>` is actual sentence entered y the user,
- `<CLASS_LABEL here>` is the class label decided by your classifier,
- `<CLASS_A>`, `<CLASS_B>` are available labels (SPAM/HAM, POSITIVE/NEGATIVE, etc.).

Classifier testing results:

Enter your classifier performance metrics below:

With ALL pre-processing steps:	Without <Lowercasing> step:
Number of true positives: 4159 Number of true negatives: 4544 Number of false positives: 138 Number of false negatives: 139 Sensitivity (recall): 0.9676 Specificity: 0.9705 Precision: 0.96788 Negative predictive value: 0.9703 Accuracy: 0.9691 F-score: 0.9677	Number of true positives: 4217 Number of true negatives: 4476 Number of false positives: 165 Number of false negatives: 122 Sensitivity (recall): 0.97188 Specificity: 0.9644 Precision: 0.9623 Negative predictive value: 0.97 Accuracy: 0.9680 F-score: 0.9670

What are your observations and conclusions? When did the algorithm perform better? a summary below

Summar / observations / conclusions

For this assignment, I utilized the Fake and Real News dataset from Kaggle to develop a classification model capable of distinguishing between fake and real news articles, text, or sentences. To speed up count and probability calculations during training and testing, I used bag of word matrices as inputs. This approach allowed for efficient matrix multiplication, matrix dot product, and broadcasting calculations, as opposed to working through data frames. To prevent underflow, I implemented add1 Laplace smoothing and used the sum of log probabilities in probability calculations for the Naive Bayes Classifier. While the classification algorithm itself was not time-consuming, stemming the train/test data proved to memory and cpu limitations.

Upon testing the test_data, the algorithm shows a high accuracy score of 96.91%. However, it struggled with colloquial language and slang. Interestingly, omitting lowercasing did not significantly impact the model's performance. Overall, the algorithm proved reliable means of categorizing news articles and tittles as either real or fake. This of course works with simple sentences as well, but may not be as accurate.