Angelica Madera
Sameeha Ahmed <- 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 STEMMING step:
Number of true positives: 872.2 Number of true negatives: 834.0 Number of false positives: 834.0 Number of false negatives: 4170.0 Sensitivity (recall): 0.47424693171705795 Specificity: 0.8340023919949953 Precision: 0.42876253743611736 Negative predictive value: 0.8407342508136996 Accuracy: 0.756581100145808 F-score: 0.14197619389072136	Number of true positives: 844.6 Number of true negatives: 861.6 Number of false positives: 861.6 Number of false negatives: 4308.0 Sensitivity (recall): 0.4706828532850295 Specificity: 0.8341881116099966 Precision: 0.4180404889808784 Negative predictive value: 0.8412548129372203 Accuracy: 0.7532575390433075 F-score: 0.14039618488628028

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

Summary / observations / conclusions
The overall accuracy of our classifier came out to ~ 75%. Meaning there was a 75% chance the classifier labeled the test reviews correctly. The metrics calculated while foregoing the preprocessing steps as well as when the preprocessing steps were ignored were pretty similar. But there was a slight improvement when the classifier performed the preprocessing steps before performing the algorithm on the data. This makes sense as preprocessing the data helps remove inconsistent words and cleans up the data, improving accuracy altogether.