Vandan Vaishya

**CS 585 Spring 2024 Programming Assignment #02**

Due: **Sunday, March 24, 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\_CS585\_Programming02.zip

1. Submit it to Blackboard Assignments section before the due date. **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. **You can work in groups of two or by yourself. Two individual students / groups can use the same data set**.

**Data set:**

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

**Deliverables:**

Your submission (if you are working as a group of two, both partners should submit the same work) should include:

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

CS585\_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. Name it:

LastName\_FirstName\_CS585\_P02\_Slides.pptx or pdf

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

LastName\_FirstName\_CS585\_P02.pdf

**Implementation:**

Your task is to implement (**from scratch – you can’t use out-of-the-box Python package classifier**), 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 CS585\_P02\_AXXXXXXXX.py TRAIN\_SIZE

where:

* + CS585\_P02\_AXXXXXXXX.py is your python code file name,
  + TRAIN\_SIZE is a number between 20 and 80 defining the size (in percentages) of the training set. For example: 60 would mean **FIRST** (as ordered in the dataset file) 60% of samples. **Note that your test set is always going to be the LAST (as ordered in the dataset file) 20% of samples.**

Example:

python CS585\_P01\_A11111111.py YES

If the number of arguments provided is NOT one (none, two or more) or the TRAIN\_SIZE argument is out of the specified range, assume that the value for TRAIN\_SIZE is 80.

* 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),
* Train your classifier on your data set:
  + assume that vocabulary V is the set of ALL words in the data set,
  + divide your data set into:
    - training set: FIRST (as they appear in the data set) TRAIN\_SIZE % of samples / documents,
    - test set: LAST 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 – but bring it back to linear space after!) 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:

Training set size: 80 %

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:

* 80 would be replaced by the value specified by TRAIN\_SIZE,
* 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 TRAIN\_SIZE set to 80: | With TRAIN\_SIZE set to 70: |
| Number of true positives: 4334  Number of true negatives: 4265  Number of false positives: 728  Number of false negatives: 673  Sensitivity (recall): 0.8655  Specificity: 0.8541  Precision: 0.8561  Negative predictive value: 0.8637  Accuracy: 0.8599  F-score: 0.8608 | Number of true positives: 6501  Number of true negatives: 6392  Number of false positives: 1098  Number of false negatives: 1009  Sensitivity (recall): 0.8656  Specificity: 0.8534  Precision: 0.8555  Negative predictive value: 0.86366  Accuracy: 0.8595  F-score: 0.8605 |

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

|  |
| --- |
| **Summary / observations / conclusions** |
| * Stable Performance Metrics: The algorithm consistently performs well, no matter how much data it's trained on. This means it's reliable. * Effect of Training Size on Correct Predictions: When we use more data (set to 70), the algorithm is better at finding both the right answers (true positives) and correctly identifying things that are not what we're looking for (true negatives). * Effect on Mistakes: With more training data (set to 70), the algorithm makes fewer mistakes in identifying things incorrectly (false positives and false negatives). * Precision and Correctness: When we train with more data (set to 70), the algorithm is a bit more precise in finding what we're looking for and in ruling out what we're not. * Overall Accuracy: The algorithm's overall performance is slightly better when it's trained with more data (set to 70). * Conclusion: Generally, the algorithm works a bit better when it's trained with 70% of the data compared to 80%. This is because it's more accurate in finding the right answers and making fewer mistakes. However, the differences between using 70% and 80% of the data are small, showing that the algorithm is still pretty reliable across different amounts of training data. |