Project 3 – Naïve Bayes Spam Detection

Implement a spam detector using the Naïve Bayes algorithm to predict if a text message is a spam:
- Use the provided python notebook to fill in your implementation
- Use the provided SMS.txt text file as the input data

Pandas DataFrame will be used as the data structure: spreadsheet manipulated through an API
- Tutorial: https://www.datacamp.com/tutorial/pandas-tutorial-dataframe-python
- Overview: https://pandas.pydata.org/docs/user_guide/10min.html
- Reference: https://pandas.pydata.org/docs/reference/frame.html

Spam Detection using Naïve Bayes

Model (w_i is a word from a text message)

$$P(class|\,message\,) \propto P(class)\prod P(w_i|class)$$

where $$P(w_i|class) = \frac{N_{wi|class}+\alpha}{N_{class}+\alpha N_{vocabulary}}$$

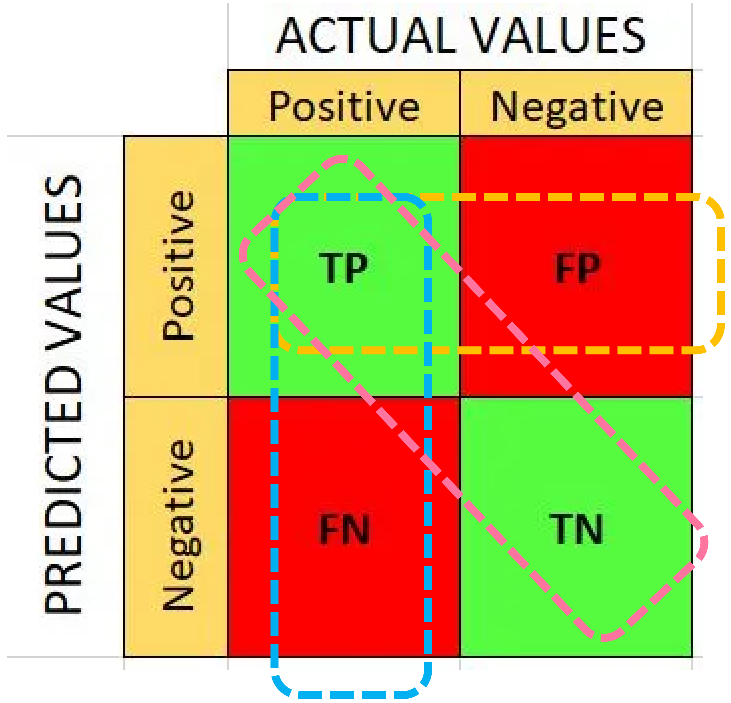
$N_{wi|class}$: The number of times the word w_i occurs in class (spam or ham) messages
N_{class}: total number of words in class messages
$N_{vocabulary}$: total number of words in the vocabulary
α: Laplace smoothing parameter to avoid the zero probability problem
(Ref: https://towardsdatascience.com/laplace-smoothing-in-na%C3%AFve-bayes-algorithm-9c237a8bdece)

A message is a spam if $P(spam|message) > P(\neg spam|message)$

CHAPMAN UNIVERSITY

Evaluating a classifier: the confusion matrix

$$\text{F1 Score} = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$



$$\text{Precision} = \frac{TP}{TP + FP} = \frac{\text{Predictions Actually Positive}}{\text{Total Predicted positive}}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{\text{Correct Predictions}}{\text{Total Predictions}}$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{\text{Predictions Actually Positive}}{\text{Total Actual positive}}$$

All evaluation scores are between 0 and 1 with 1 being the best

CHAPMAN UNIVERSITY

Submission:

You must use the provided Python notebook and add your implementation code to the specified areas. Submit one notebook file containing your implementation code, output, and documentation. Provide ample comments to explain your implementation.

Output:

1. A summary of your prediction results in a confusion matrix.
2. Details of the probability calculation (i.e., the probabilities for spam and ham of each word) for one example from each quadrant of the confusion matrix
3. Analysis of why those examples are classified correctly or incorrectly.

Each student should complete the project individually.

You can use utility packages and dataframe functions, but you must implement the core algorithm for probability calculation and confusion matrix aggregation.

Grading:

- 50% for successful execution and correct output
- 30% for the prediction performance of your implementation
- 20% for the detailed examples and your analysis
- Extra 20% if you also implement the detector using MultinomialNB from sklearn and compare its performance