Group 11 (19CS10071, 19CS30041)
Anurat Bhattacharya
Sayantan Saha

Machine Learning Assign 2

19th October 2021

OVERVIEW

The task is to implement and use a Naive Bayes Classifier for Spooky Author Identification from the database https://www.kaggle.com/c/spooky-author-identication/overview and to compute 95% confidence of accuracy, precision. f-score, sensitivity and specificity of the model.

GOALS

1. First, we need to read the data.
2. Then, the data read is prepared to be fed for the implementation like parsing of the matrix, train-test split etc.
3. Analyse the results (95% confidence of accuracy, precision. f-score, sensitivity and specificity of the model)

Description of the dataset

The dataset given to us contains following attributes:

1. id: It contains the id of the respective texts in the long dataset.
2. text: It contains the raw text corresponding to the id. This text field is used to create the feature matrix.
3. author: This field contains the name of the author corresponding to the text. There are only three different authors in the entire dataset. The three authors are : EAP, HPL and MWS.

Our aim is to use the 'text' feature as input attributes and predict the author from the words present in the text using a sparse matrix of word vectors from a Naive Bayes Classifier model .

Steps to Run:

1.Run pip install -r requirements.txt

2.Run python assgn2_19CS10071_19CS30041.py

PROCEDURE

1. We will tokenize the words from the texts given in the ‘text’ field of the given dataset. Then the stopwords are avoided from that tokenized list.
2. So, the feature matrix M is created which is a r ×c binary feature matrix where r is the number of examples and c is the size of the vocabulary consisting of distinct words present in the dataset. Each row corresponds to an example of the dataset and each column corresponds to a word in the vocabulary. $M_{ij} = 1$ if and only if the j-th word is present in the text of the i-th example.
3. As we found that the matrix M is large and sparse, we use the scipy csr matrix to represent it and make a train test split of 70:30.
4. Hence, the data is prepared. Thereafter, from the training set, the prior probabilities are calculated of each word vector given each class/example. And the likelihoods are also calculated for the training dataset.
5. Now we can create the model using the prior and likelihood to find the maximum posterior hypothesis.
6. We have to analyse with and without Laplace correction. While we analyse without laplace correction we can’t use log-likelihood. So, we use the normal non-log results for computing the results for the 2nd question because we can face a log(0) which is undefined. But, while applying Laplace correction, there is no such case. So, we can use log-likelihood there.For laplace correction, we use the formula, $\frac{p+\alpha}{N+\alpha N}$,to prevent vanishing likelihoods and here we take alpha = 1.
7. Then, we analyse the result with 95% confidence of accuracy, precision. f-score, sensitivity and specificity of the model. Now to analyse these results, we use a 3 ×3 confusion matrix for the three authors. As precision, f-score, sensitivity and specificity are binary features we need to calculate pairwise results for the three authors.

Implementation of the Naive Bayes Classifier:

Description of some major functions:

1. **get_word_tokens(text):** This function takes a raw text (‘text’) as a parameter and tokenizes the words existing in that text. Stopwords are removed in this function.
2. **train_test_split(tdm,y):** This function takes the sparse matrix of the independent features (‘tdm’), corresponding set of values for the dependent feature (‘y’) and splits both of them in training and test data. The split is 70:30 for train and test respectively.
3. **prep_data(data_):** This function prepares data from the data read by tokenizing the works in the entire dataset (‘dataset_’) other than the stopwords, then parsing the feature matrix and splits the data in training and testing set using previously described functions. It returns the word vector consisting of the dictionary mapping word tokens to their unique individual ids.
4. **calc_likelihood(y:pd.DataFrame,word_vector:dict,tdm:csr_matrix):** The parameter ‘y’ represents the set of values for he dependent feature, ‘tdm’ represents the sparse matrix for the independent features, ‘word_vector’ represents the dictionary of all the words present in the entire dataset. This function returns a dictionary of all the likelihood of each word vector given each class/example.
5. **calc_priors(y:pd.DataFrame):** Here in this function y is as follows from the previous function. It returns a dictionary containing the prior probabilities.
6. **class model :** Class used to do combine together everything and do the training and performance analysis
7. **model::predict(x) :** It returns the predicted author for the given input binary word_vector row x_.
8. **model::eval() :** It calculates and prints the Accuracy with 95%CI, precision,sensitivity ,specificity and f-score.

Analysing the Model Performance:

- **95% confidence interval of the accuracy:** The 95% confidence interval is a range of values that one can be 95% confident contains the true accuracy of the model. Due to natural sampling variability, the mean accuracy (center of the confidence interval) will vary from sample to sample.Mean Estimate of the actual accuracy is equal to the current model accuracy.

Confidence Interval formula: $\mu \pm z.\sigma$ where μ is the sample mean of accuracy, z is the chosen Z-value(1.96 for 95% confidence interval) and σ is the sample standard deviation of accuracy.

Approximating the accuracy as a normal distribution with mean(=accuracy got from the current model), variance(((accuracy*(1-accuracy))/n), where n is length of validation set and z = 1.96 for 95% CI.

Precision: The probability of the observed result arising by chance. The precision-value is the chance of getting the reported study result (or one even more extreme) when the null hypothesis is actually true. Thus, a very small precision-value gives rise to the impression that the results are valid, because it suggests that the investigator's hypothesis was in fact completely wrong.

So, precision is the fraction of true positive examples among the examples that the model classified as positive. In other words, the number of true positives divided by the number of false positives plus true positives.

- **F-score:** The F-score, also called the F1-score, is a measure of a model's accuracy on a dataset. It is used to evaluate binary classification systems, which classify examples into 'positive' or 'negative'.

$$\text{F-score} = \frac{2}{(presision^{-1} + sensitivity^{-1})} = \frac{tp}{tp + \frac{1}{2}(fp + fn)}.$$

tp: The number of true positives classified by the model.
fp: The number of false positives classified by the model.
fn: The number of false negatives classified by the model.

- **Sensitivity:** Sensitivity (True Positive Rate) refers to the proportion of those who received a positive result on this test out of those who actually have the condition.

$$\text{Sensitivity} = \frac{tp}{tp + fn}.$$

tp: The number of true positives classified by the model.
fn: The number of false negatives classified by the model.

- **Specificity:** Specificity (True Negative Rate) refers to the proportion of those who received a negative result on this test out of those who do not actually have the condition.

$$\text{Specificity} = \frac{tn}{tn + fp}$$

tn: The number of true negatives classified by the model.

fp: The number of false positives classified by the model.

Results Obtained :

1.Without Laplace Correction

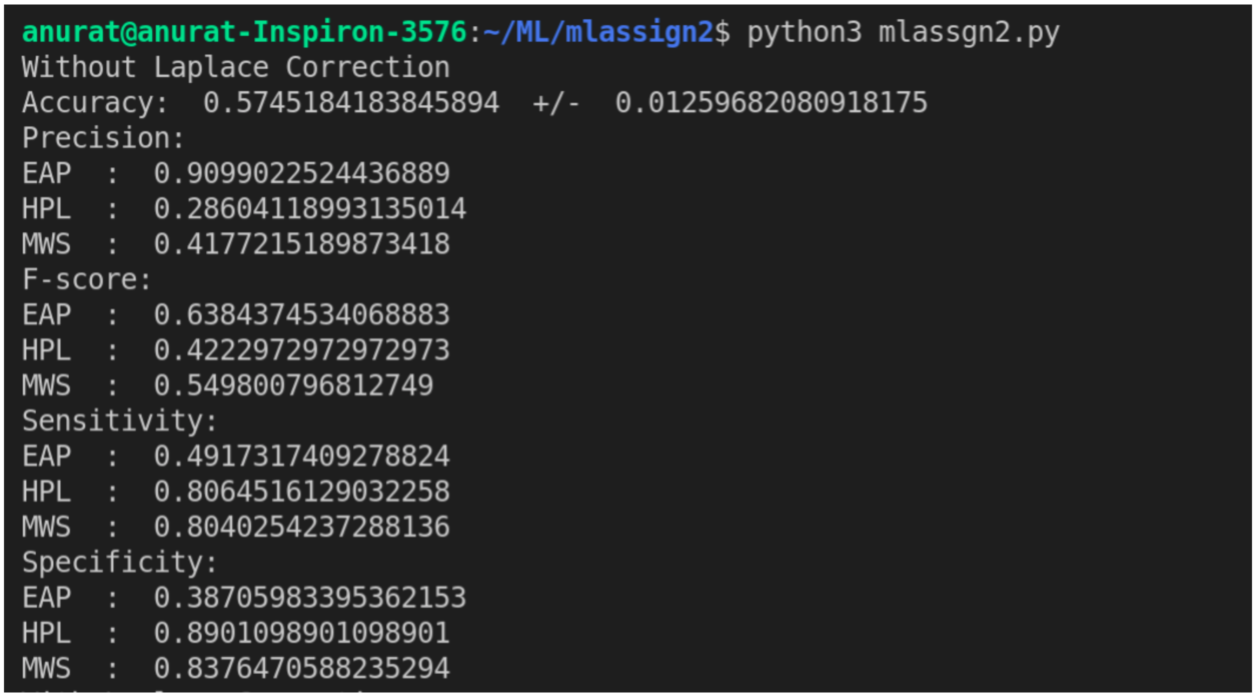
Accuracy: 0.5745184183845894 +/- 0.01259682080918175

Precision:
EAP : 0.9099022524436889
HPL : 0.28604118993135014
MWS : 0.4177215189873418

F-score:
EAP : 0.6384374534068883
HPL : 0.4222972972972973
MWS : 0.549800796812749

Sensitivity:
EAP : 0.4917317409278824
HPL : 0.8064516129032258
MWS : 0.8040254237288136

Specificity:
EAP : 0.38705983395362153
HPL : 0.8901098901098901
MWS : 0.8376470588235294


```
anurat@anurat-Inspiron-3576:~/ML/mlassign2$ python3 mlassgn2.py
Without Laplace Correction
Accuracy:  0.5745184183845894  +/-  0.01259682080918175
Precision:
EAP  :  0.9099022524436889
HPL  :  0.28604118993135014
MWS  :  0.4177215189873418
F-score:
EAP  :  0.6384374534068883
HPL  :  0.4222972972972973
MWS  :  0.549800796812749
Sensitivity:
EAP  :  0.4917317409278824
HPL  :  0.8064516129032258
MWS  :  0.8040254237288136
Specificity:
EAP  :  0.38705983395362153
HPL  :  0.8901098901098901
MWS  :  0.8376470588235294
```

2.With Laplace Correction
Accuracy: 0.8306370070778564 +/- 0.009543269718694807

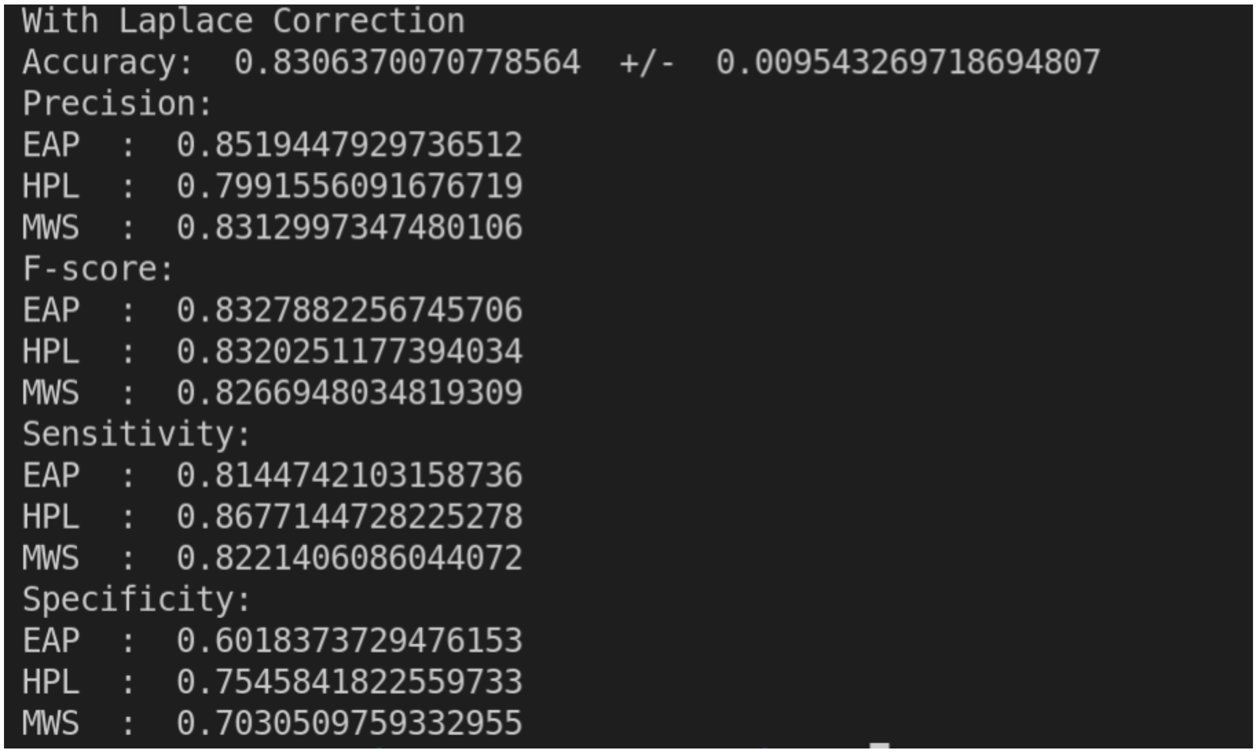
Precision:
EAP : 0.8519447929736512
HPL : 0.7991556091676719
MWS : 0.8312997347480106

F-score:
EAP : 0.8327882256745706
HPL : 0.8320251177394034
MWS : 0.8266948034819309

Sensitivity:
EAP : 0.8144742103158736
HPL : 0.8677144728225278
MWS : 0.8221406086044072

Specificity:
EAP : 0.6018373729476153
HPL : 0.7545841822559733

MWS : 0.7030509759332955



```
With Laplace Correction
Accuracy:  0.8306370070778564  +/-  0.009543269718694807
Precision:
EAP  :  0.8519447929736512
HPL  :  0.7991556091676719
MWS  :  0.8312997347480106
F-score:
EAP  :  0.8327882256745706
HPL  :  0.8320251177394034
MWS  :  0.8266948034819309
Sensitivity:
EAP  :  0.8144742103158736
HPL  :  0.8677144728225278
MWS  :  0.8221406086044072
Specificity:
EAP  :  0.6018373729476153
HPL  :  0.7545841822559733
MWS  :  0.7030509759332955
```