PROJECT 5

Course: CSE 537

Due date: December 8, 2018

TEAM MEMBER 1:
Name: Ayushi Srivastava
Student Id: 112101239
Email: ayushi.srivastava@stonybrook.edu

TEAM MEMBER 2:
Name: Jahnavi Tharigopula
Student Id: 112078393
Email: jahnavi.tharigopula@stonybrook.edu

Que 1.

We are required to implement the ID3 decision tree that uses the chi-squared split stopping criterion with different p-value threshold given as a parameter. We have run our implementation with different p-values and below are the results.

- **For p=0.01**

  python q1_classifier.py -p 0.01 -f1 train.csv -f2 test.csv -o output.csv -t tree.pkl
  Data Loading: done
  Training...
  Testing...
  Output files generated
  ('No.of Internal Nodes: ', 26)
  ('No.of Leaf Nodes: ', 105)
  ('Total No.of Nodes: ', 131)

And on running the autograder

  python autograder_basic.py
  Data Loading: done
  Tree prediction accuracy: 0.75216
  Output file prediction accuracy: 0.75216
  Tree prediction matches output file

- **For p=0.05**

  python q1_classifier.py -p 0.05 -f1 train.csv -f2 test.csv -o output.csv -t tree.pkl
  Data Loading: done
  Training...
  Testing...
  Output files generated
  ('No.of Internal Nodes: ', 35)
  ('No.of Leaf Nodes: ', 141)
  ('Total No.of Nodes: ', 176)

And on running the autograder

  python autograder_basic.py
  Data Loading: done
  Tree prediction accuracy: 0.75324
  Output file prediction accuracy: 0.75324
  Tree prediction matches output file

- **For p=1**

  python q1_classifier.py -p 1 -f1 train.csv -f2 test.csv -o output.csv -t tree.pkl
  Data Loading: done

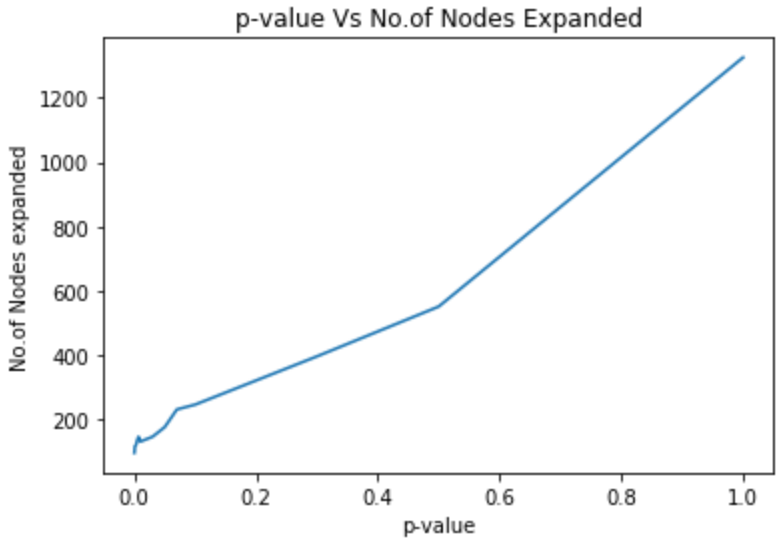
Training...
Testing...
Output files generated
('No.of Internal Nodes: ', 265)
('No.of Leaf Nodes: ', 1061)
('Total No.of Nodes: ', 1326)

And on running the autograder

- python autograder_basic.py
  Data Loading: done
  Tree prediction accuracy: 0.74832
  Output file prediction accuracy: 0.74832
  Tree prediction matches output file

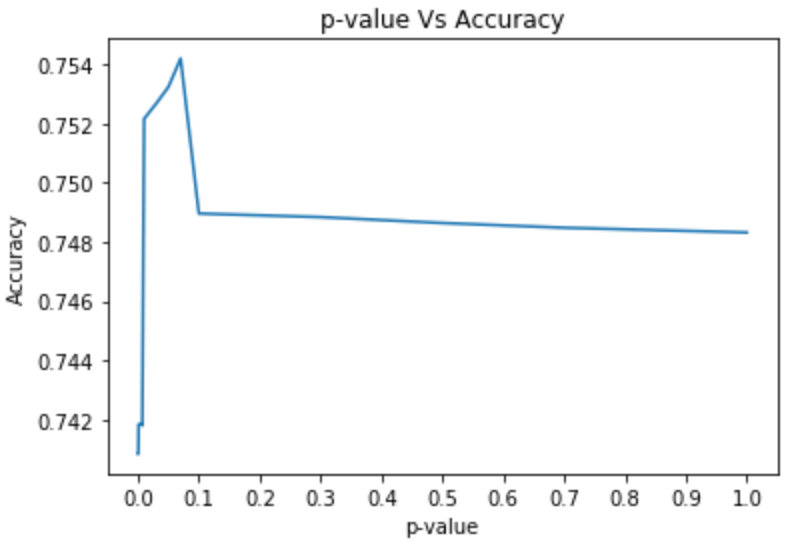
Observations:

- We have run our code for the following p-values and plotted it against the total no.of nodes expanded and we clearly see that the no.of nodes being expanded increase as we increase the p-value.
  - **<u>P-values :</u> 0.0001, 0.0005, 0.001, 0.003, 0.005, 0.007, 0.01, 0.03, 0.05, 0.07, 0.1, 0.3, 0.5, 0.7, 1**
  - The results for all of the p-values have been reported at the end of the report.



- For p=1, the entire tree is expanded, hence the no.of nodes is highest and for lower p-value, many nodes are pruned, thus have lesser no.of nodes expanded.

- We also plotted the accuracies for all the above tests
  - **<u>P-values :</u> 0.0001, 0.0005, 0.001, 0.003, 0.005, 0.007, 0.01, 0.03, 0.05, 0.07, 0.1, 0.3, 0.5, 0.7, 1**
  - The results for all of the p-values have been reported at the end of the report.



- For higher p-value p=1, the entire tree is expanded without any pruning and this may have resulted in overfitting, hence causing the decrease in the accuracy.
- And for lower p-values, excessive pruning may have taken place, thus resulting in underfitting, hence causing the lower accuracies.
- There exists an optimal p-val in between 0 and 1 around **pval = 0.05** where there is no over or underfitting and the accuracy is the highest.

Que 2

Required to implement the spam classifier using Naive Bayes technique.

We first implemented the spam classifier without any feature extraction and the results are provided below:

python q2_classifier.py -f1 train.csv -f2 test.csv -o testoutput_2
Accuracy of the result is 90.2 %
Precision spam: 87.77%
Precision ham: 94.48%
Recall: 96.55 %
fmeasure: 91.95 %

Extra credit question:

Then, the given corpus in the question, extracted the stop_words from those mails.
Stop words are the words which are used frequently in common English language and also don't contribute to any spam /non-spam mails. Eg - ("the", "of", "there", etc.). These words won't make any difference for classifying our mails and hence if we remove these words , our results will be more accurate. So, after identifying these words from the given mails , got the accuracy of 91.5%

python q2_classifier.py -f1 train.csv -f2 test.csv -o testoutput_2
Accuracy of the result is 91.5 %
Precision spam: 87.67%
Precision ham: 98.83%
Recall: 99.31%
fmeasure: 93.13 %

Some points to consider :

- We tried various smoothing parameters:

1. **Additive smoothing** - If we encounter a word in the dataset, which is not a part of the dataset, so we get its probability as 0. So, to tackle this issue, we introduce additive smoothing, where we make sure none of the probabilities are 0. In additive smoothing we add a number alpha to the numerator and add alpha times number of classes over which the probability is found in the denominator.

   Results for different smoothing values:
   - **Smoothing value - 5**
     Accuracy of the result is 91.3 %
     Precision of spam is: 87.40%
     Precision of not spam is: 98.83 %
     Recall parameter is: 99.31 %
     fmeasure is : 92.97%

   - **Smoothing value - 10**
     python q2_classifier.py -f1 train -f2 test -o out
     Accuracy of the result is 91.2 %
     Precision of spam is: 87.27%
     Precision of not spam is: 98.82 %
     Recall parameter is: 99.31 %
     fmeasure is : 92.90 %

2. **Laplace smoothing** - If alpha is 1 in above, it is **laplace smoothing.**
   python q2_classifier.py -f1 train -f2 test -o out
   Accuracy of the result is 91.5 %
   Precision of spam is: 87.67 %
   Precision of not spam is: 98.83 %
   Recall parameter is: 99.31 %
   fmeasure is : 93.13%

Observations:

- **Underflow Prevention** - Multiplying lots of probabilities, which are between 0 and 1 by definition, can result in floating-point underflow. • Since log(xy) = log(x) + log(y), it is better to perform all computations by summing logs of probabilities rather than multiplying probabilities. We get the Class with highest final un-normalized log probability score is still the most probable.
  Therefore, used - np.log10(self.laplace_smooth / smoothing) to calculate the probability after applying smoothing.
- Increased the accuracy by removing **stop_words.** Stop words are those words which occur extremely frequently in any text. For example words like, 'the', 'a', 'an', etc. These words do not give us any information about the content of the text. Thus it should not matter if we remove these words from the text. These words were extracted from the given corpus in the 'Original files' of Sahami's paper. And put into the code for better and accurate results of detecting the spam. More words can be added in the stop words to increase its accuracy.
- One more method to improve accuracy, we can use - **n- grams** - Instead of counting single words, we can count group of words

Results for Que 1 for different P-values:

P = 0.0001

```
python q1_classifier_changed.py -p 0.0001 -f1 train.csv -f2 test.csv -o output.csv -t tree.pkl
Data Loading: done
Training...
Testing...
Output files generated
('No.of Internal Nodes: ', 19)
('No.of Leaf Nodes: ', 77)
('Total No.of Nodes: ', 96)

python autograder_basic.py
Data Loading: done
Tree prediction accuracy:  0.74084
Output file prediction accuracy:  0.74084
Tree prediction matches output file
```

P = 0.0005

```
python q1_classifier.py -p 0.0005 -f1 train.csv -f2 test.csv -o output.csv -t tree.pkl
Data Loading: done
Training...
Testing...
Output files generated
```

('No.of Internal Nodes: ', 20)
('No.of Leaf Nodes: ', 81)
('Total No.of Nodes: ', 101)

python autograder_basic.py
Data Loading: done
Tree prediction accuracy: 0.74084
Output file prediction accuracy: 0.74084
Tree prediction matches output file

P = 0.001
python q1_classifier_changed.py -p 0.001 -f1 train.csv -f2 test.csv -o output.csv -t tree.pkl
Data Loading: done
Training...
Testing...
Output files generated
('No.of Internal Nodes: ', 23)
('No.of Leaf Nodes: ', 93)
('Total No.of Nodes: ', 116)

python autograder_basic.py
Data Loading: done
Tree prediction accuracy: 0.74184
Output file prediction accuracy: 0.74184
Tree prediction matches output file

P = 0.003
python q1_classifier_changed.py -p 0.003 -f1 train.csv -f2 test.csv -o output.csv -t tree.pkl
Data Loading: done
Training...
Testing...
Output files generated
('No.of Internal Nodes: ', 24)
('No.of Leaf Nodes: ', 97)
('Total No.of Nodes: ', 121)

python autograder_basic.py
Data Loading: done
Tree prediction accuracy: 0.74184
Output file prediction accuracy: 0.74184
Tree prediction matches output file

P = 0.005

```
python q1_classifier_changed.py -p 0.005 -f1 train.csv -f2 test.csv -o output.csv -t tree.pkl
Data Loading: done
Training...
Testing...
Output files generated
('No.of Internal Nodes: ', 27)
('No.of Leaf Nodes: ', 109)
('Total No.of Nodes: ', 136)

python autograder_basic.py
Data Loading: done
Tree prediction accuracy:  0.74184
Output file prediction accuracy:  0.74184
Tree prediction matches output file
```

P = 0.007

```
python q1_classifier_changed.py -p 0.007 -f1 train.csv -f2 test.csv -o output.csv -t tree.pkl
Data Loading: done
Training...
Testing...
Output files generated
('No.of Internal Nodes: ', 29)
('No.of Leaf Nodes: ', 117)
('Total No.of Nodes: ', 146)

python autograder_basic.py
Data Loading: done
Tree prediction accuracy:  0.7418
Output file prediction accuracy:  0.7418
Tree prediction matches output file
```

P = 0.01

```
python q1_classifier_changed.py -p 0.01 -f1 train.csv -f2 test.csv -o output.csv -t tree.pkl
Data Loading: done
Training...
Testing...
Output files generated
('No.of Internal Nodes: ', 26)
('No.of Leaf Nodes: ', 105)
('Total No.of Nodes: ', 131)
```

python autograder_basic.py
Data Loading: done
Tree prediction accuracy: 0.75216
Output file prediction accuracy: 0.75216
Tree prediction matches output file

P = 0.03
python q1_classifier_changed.py -p 0.03 -f1 train.csv -f2 test.csv -o output.csv -t tree.pkl
Data Loading: done
Training...
Testing...
Output files generated
('No.of Internal Nodes: ', 29)
('No.of Leaf Nodes: ', 117)
('Total No.of Nodes: ', 146)

python autograder_basic.py
Data Loading: done
Tree prediction accuracy: 0.75268
Output file prediction accuracy: 0.75268
Tree prediction matches output file

P = 0.05
python q1_classifier_changed.py -p 0.05 -f1 train.csv -f2 test.csv -o output.csv -t tree.pkl
Data Loading: done
Training...
Testing...
Output files generated
('No.of Internal Nodes: ', 35)
('No.of Leaf Nodes: ', 141)
('Total No.of Nodes: ', 176)

python autograder_basic.py
Data Loading: done
Tree prediction accuracy: 0.75324
Output file prediction accuracy: 0.75324
Tree prediction matches output file

P = 0.07
python q1_classifier.py -p 0.07 -f1 train.csv -f2 test.csv -o output.csv -t tree.pkl
Data Loading: done
Training...
Testing...

Output files generated
('No.of Internal Nodes: ', 46)
('No.of Leaf Nodes: ', 185)
('Total No.of Nodes: ', 231)

python autograder_basic.py
Data Loading: done
Tree prediction accuracy: 0.7542
Output file prediction accuracy: 0.7542
Tree prediction matches output file

P = 0.1
python q1_classifier_changed.py -p 0.1 -f1 train.csv -f2 test.csv -o output.csv -t tree.pkl
Data Loading: done
Training...
Testing...
Output files generated
('No.of Internal Nodes: ', 49)
('No.of Leaf Nodes: ', 197)
('Total No.of Nodes: ', 246)

python autograder_basic.py
Data Loading: done
Tree prediction accuracy: 0.74896
Output file prediction accuracy: 0.74896
Tree prediction matches output file

P = 0.3
python q1_classifier.py -p 0.3 -f1 train.csv -f2 test.csv -o output.csv -t tree.pkl
Data Loading: done
Training...
Testing...
Output files generated
('No.of Internal Nodes: ', 79)
('No.of Leaf Nodes: ', 317)
('Total No.of Nodes: ', 396)

python autograder_basic.py
Data Loading: done
Tree prediction accuracy: 0.74884
Output file prediction accuracy: 0.74884
Tree prediction matches output file

P = 0.5

```
python q1_classifier_changed.py -p 0.5 -f1 train.csv -f2 test.csv -o output.csv -t tree.pkl
Data Loading: done
Training...
Testing...
Output files generated
('No.of Internal Nodes: ', 110)
('No.of Leaf Nodes: ', 441)
('Total No.of Nodes: ', 551)

python autograder_basic.py
Data Loading: done
Tree prediction accuracy:  0.74864
Output file prediction accuracy:  0.74864
Tree prediction matches output file
```

P = 0.7

```
python q1_classifier.py -p 0.7 -f1 train.csv -f2 test.csv -o output.csv -t tree.pkl
Data Loading: done
Training...
Testing...
Output files generated
('No.of Internal Nodes: ', 172)
('No.of Leaf Nodes: ', 689)
('Total No.of Nodes: ', 861)

python autograder_basic.py
Data Loading: done
Tree prediction accuracy:  0.74848
Output file prediction accuracy:  0.74848
Tree prediction matches output file
```

P = 1

```
python q1_classifier_changed.py -p 1 -f1 train.csv -f2 test.csv -o output.csv -t tree.pkl
Data Loading: done
Training...
Testing...
Output files generated
('No.of Internal Nodes: ', 265)
('No.of Leaf Nodes: ', 1061)
('Total No.of Nodes: ', 1326)
```

```
python autograder_basic.py
Data Loading: done
Tree prediction accuracy:  0.74832
Output file prediction accuracy:  0.74832
Tree prediction matches output file
Jahnavis-MacBook-Pro:Project5 jahnavichowdary$
```