Spam Filtering – Naïve Bayes vs kNN

By: Namrata Kolla

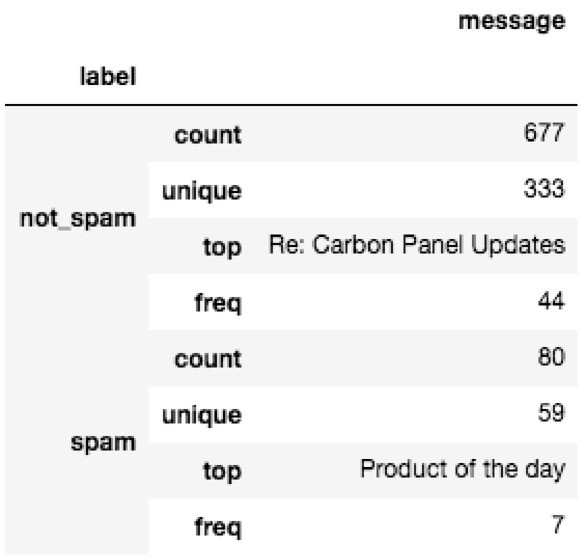
Date: February 21, 2018

---

1. Execution and Performance Metrics

1.1. Introduction

This report compares the effectiveness of using Naive Bayes and k-Nearest-Neighbors algorithms to classify email as spam. The analysis fed 757 email subjects into the algorithms. Roughly 10% of the email subjects are spam, and spam subjects are more likely to be unique than non-spam subjects:



		message
label		
not_spam	**count**	677
	unique	333
	top	Re: Carbon Panel Updates
	freq	44
spam	**count**	80
	unique	59
	top	Product of the day
	freq	7

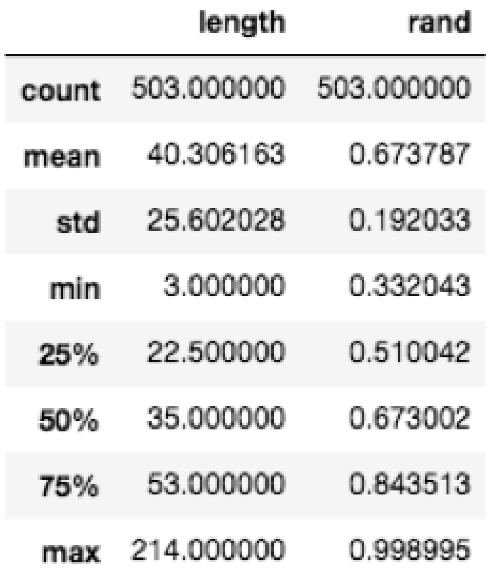
** Note: More than 757 emails should be collected in the future to improve the accuracy of these models. Because of the time it took to import my email data, I only collected 757 for this report... and it still performs well!

1.2. Feature Engineering and Splitting

This report takes a bag-of-words approach in engineering the features. The email subject strings were turned into vectors based on how many times each word occurred in the message (term frequency), weighting the counts so that frequent tokens got lower weights (inverse document frequency), and normalizing the vectors to unit length (L2 norm). Each vector ends up with as many dimensions as there are unique words in the subject.

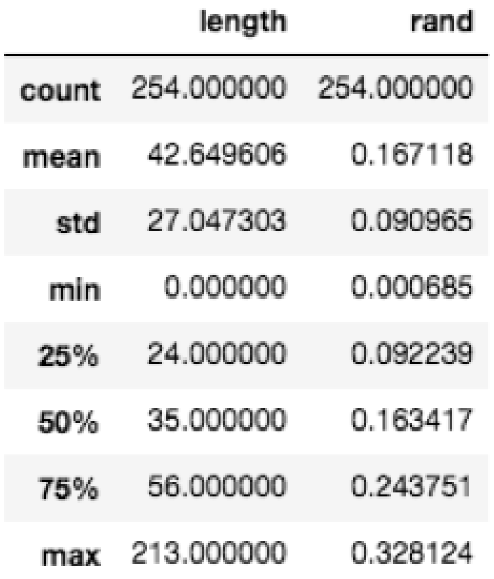
Then the data was split where 66% was used for training and 33% was used for testing:

TRAINING



	length	rand
count	503.000000	503.000000
mean	40.306163	0.673787
std	25.602028	0.192033
min	3.000000	0.332043
25%	22.500000	0.510042
50%	35.000000	0.673002
75%	53.000000	0.843513
max	214.000000	0.998995

TESTING

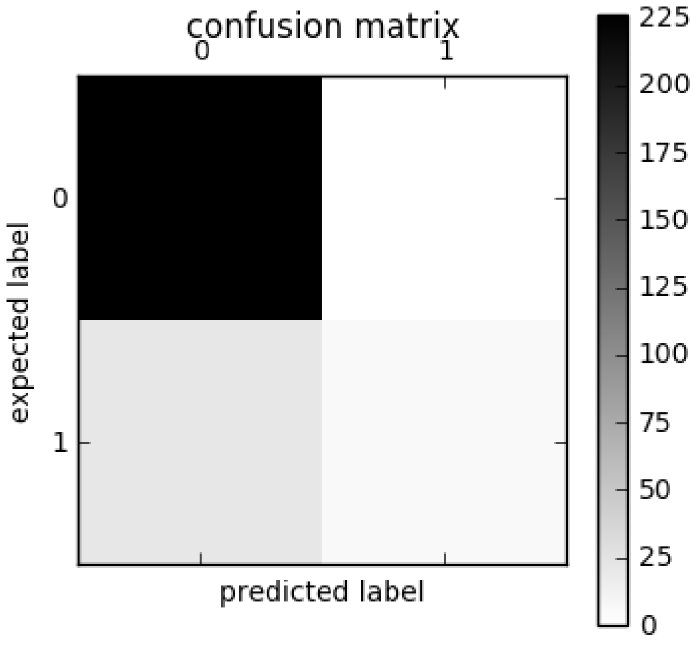


	length	rand
count	254.000000	254.000000
mean	42.649606	0.167118
std	27.047303	0.090965
min	0.000000	0.000685
25%	24.000000	0.092239
50%	35.000000	0.163417
75%	56.000000	0.243751
max	213.000000	0.328124

1.3. Naïve-Bayes

The first algorithm used in this report, Naïve-Bayes, calculates the probability of an email subject belonging in the spam category by calculating individual probabilities for whether each word in the subject belongs in the spam category. Note that this algorithm assumes that the words are conditionally independent (having one word does not change the probability of the existence of another word). While this assumption is not entirely true, the classifier still does a good job as suggested by an accuracy score of 0.913.

The confusion matrix is very dark in the top left and very light in the top right, which suggests the algorithm did well. The dark box indicates that most of the true negatives (truly not-spam emails) were captured, and the light box indicates that most of the false negatives (predicted spam when it wasn't) were not captured. For this use case, we want a near-zero false negative rate because most people would much rather receive spam than have an important email go to their spam.



Correspondingly:

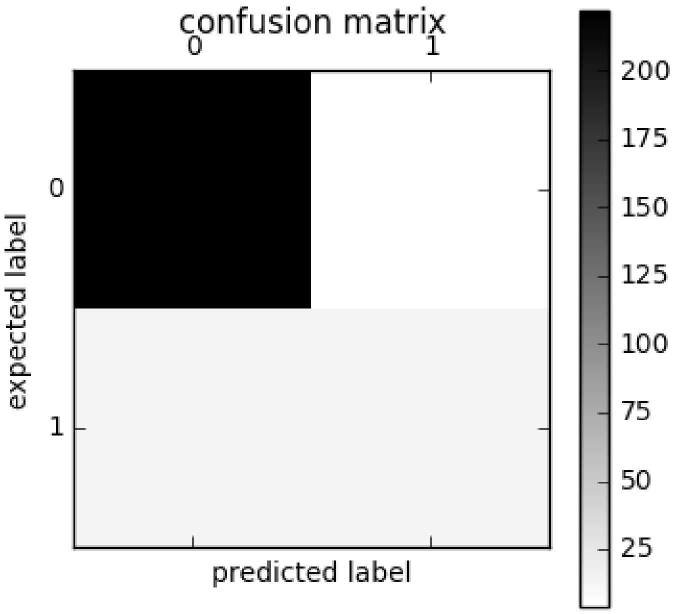
```
             precision    recall  f1-score   support

   not_spam       0.91      1.00      0.95       226
       spam       1.00      0.21      0.35        28

avg / total       0.92      0.91      0.89       254
```

The precision of both identifying spam and not_spam is high, which means our results are very useful. The recall for not_spam is perfect (all email that is not spam is captured); but the recall for spam is low where not all spam is correctly classified. The average of the f1-score is 0.89, which is slightly lower than the analogous accuracy score. The AUC score on the test data was 0.61.

1.4. k-Nearest Neighbors (kNN)

The second method used in this report, kNN, stores all the training data in a memory structure and uses them directly for classification. Each email subject is represented as a point in a space with as many dimensions as the number of features (in this case, 616). After evaluating the size of the training set, I validated that 3 is the best choice for the hyper-parameter k. Thus, this algorithm classified each email subject as spam or not_spam based on the majority class of its 3 closest neighbors.

The method using kNN had an accuracy score of 0.929. Similar to the Naive Bayes' confusion matrix, most of the true negatives (truly not-spam emails) were captured, and most of the false negatives (predicted spam when it wasn't) were not captured.



Correspondingly:

```
             precision    recall  f1-score   support

   not_spam       0.94      0.98      0.96       226
       spam       0.78      0.50      0.61        28

avg / total       0.92      0.93      0.92       254
```

The precision of identifying not_spam is high while identifying spam is not as high. This means the model's ability to identify not_spam is useful while its ability to identify spam is not as useful. The recall for not_spam is high (almost all email that is not spam is captured) and once again the recall for spam is low where 50% of the spam is misclassified. The average of the f1-score is 0.92, which is about the same as the analogous accuracy score. The AUC score on the test data was 0.74.

2. Evaluation

2.1. Performance

The biggest findings in this study are that both learning methods have a very accurate classification and kNN performed better than Naïve-Bayes: Naive Bayes had a 91.3% accuracy and kNN had a 92.9% accuracy. This is unexpected because there were 616 features (more than the number of features in the dataset ~500), and kNN tends to perform poorly with a large number of features.

kNN also outperforms Naive Bayes with regard to AUC score, which is a measure of the area under a curve of true positives vs false positives. In both cases, the true positive rate (or recall) is higher than the false positive rate (or fall-out) and the classifiers do better than randomly flipping a coin (which would have an AUC of 0.5). However, we observe that kNN does better than Naive Bayes because its AUC is closer to 1 (there is more area under the curve). This observation makes sense given that AUC depends on the measure of recall, which kNN has a higher average (0.93) of than Naive Bayes (0.91). Similar to AUC scores, the f1-score which is a combination of precision and recall, is higher for kNN than for Naive Bayes.

A third finding is that the average precision for both approaches is about the same, but Naive Bayes classifies spam better than not_spam and kNN classifies not_spam better than spam. Given that kNN takes the “average” of neighbors and spam had much fewer neighbors in this dataset than not_spam (~10% of the dataset was spam), this observation follows that it is easier to correctly identify emails as not_spam via the average of neighbors than it is to identify spam.

Overall, kNN performs better than Naïve Bayes with regard to precision, recall, f1-score and accuracy. However, we cannot expect the same outcome as more data and more features are added.

2.2. Other strengths and weaknesses & the future

In the future, this study should be repeated with more email data to improve performance of both models. One potential weakness that may arise with additional data is that kNN will take longer to classify test data. In the real world, I would recommend using Naive Bayes for its speed. kNN also has a tendency to over-fit (similar to decision trees) as data are added, which would reduce its generalizability to other email boxes.

It should also be noted that Naive Bayes makes the assumption that all features are conditionally independent. With the features used in this algorithm (bag of words approach), this assumption does not reduce performance significantly. However, if more features were to be added in the future, the issue of conditional independence should be taken more seriously. For example, if we included email send times and an email subject including the word "news", there is dependence from the fact that a lot of news-based emails are sent in the morning.