DS7333 Quantifying the World: Case Study 3
Rick Fontenot, Feby Thomas Cheruvathoor

1 Introduction

The goal for this case study is to create a spam email filter by predicting the classes "spam" or "not spam" using a naïve bayes model. In addition, a clustering model will be evaluated to see if clustering features can improve the model's performance.

Based on the performance metrics, the model's decision boundary can be shifted to the clients' preferences to balance the accuracy or to filter out as much spam as possible without discarding any truly important emails.

2 Methods

Data description

The raw data set contains 9,353 observations of email documents separated into 5 folders of classification subsets that will be grouped into the following for binary classification:

Folder	#Emails	Target Classification
easy_ham	5052	Not Spam
easy_ham_2	1401	Not Spam
hard_ham	501	Not Spam
spam	1001	Spam
spam_2	1398	Spam

Figure 1: Summary of email document categories

Each email is a Multipurpose Internet Mail Extensions (MIME) file which in addition to the email body text also has a header including information such as "To:", "From", "Subject", "Content-Type". Some of the messages are also multipart messages which may need additional processing to capture the appropriate part of the messages for modeling and predictions.

Reading files and creating initial dataset

Looping through and reading each file, an initial dataframe was created with the following features:

file_id : contains a string with the filenames to be used for identification

subject : using the "BytesParser "tool from python's "email.parser" the attribute ['Subject'] was parsed from the header and this feature contains the text of the subject

body : using python's email.message_from_file tool along with the get_payload tool, the text of body was extracted. Using the msg.walk() tool a loop through each part of each message was performed so that only parts with get_content_type() == 'text/plain' were extracted

Is_Spam : this target variable was encoded as 0 for files in folders containing "Not Spam" emails and encoded as 1 for files in folders containing "Spam" emails

After creating this initial dataset the strings for end of line and tabs (\n and \t) were removed for all records in the subject and body fields.

Cross Validation

In order to compare the models for overfitting, we first set aside 10% of the data into a validation set (stratified shuffled data to get random observations with similar proportion of records in the target classes).

For the training data used to build models, we used 10-Fold cross validation and the mean accuracy across the folds for model assessment. In addition to accuracy, we will also summarize the precision and recall for each class, for comparison.

We performed the validation split prior to further data processing and feature creation so that the process is valid to make predictions unseen data as a spam filter should. All subsequent vectorization and clustering can be computed on unseen data without re-training models.

Processing data and Feature Creation

Vectorization of documents

Python's TfidfVectorizer tool was used to transform the text into a Term Frequency Inverse Document Frequency vector. A column for each term that exists in at least one message is created where a larger weight of the tf-idf calculation represents a high term frequency(tf) in the given document(local parameter) and a low document frequency of the term in the whole collection (global parameter)

The tf-idf-vector matrices were calculated separately for the "subject" and "body" fields rather than just the body or the combination of both fields. This allows models to be assessed for the predictive power of just the subject or just the body. Then features of each matrices were concatenated into a model using both the tf-idf for body and the tf-idf for the subject.

These vectorizers were fit on the training data only, then the training fit was used to transform the validation set. When new emails come in, this same transformation can be performed without re-fitting the model.

Within our tf-idf vectorization, the following model parameters were used:

use_idf = True : to enable inverse-document-frequency reweighting

stop_words = None : it's possible the use of stop words in spam emails could be different than their usage in non-spam emails. Since the vectorization and the naïve bayes models are relatively quick compute times, these words were kept as opposed to reducing features

norm =None : no normalization of the individual vector elements was performed

ngram_range = (1,3) : in addition to creating features for individual words, features based on 2-word and 3-word phrases were also created

Clustering

After vectorization of the subject and body documents, DB-Scan clustering was performed using all term weights. The clusters are then used as additional features for models to predict spam.

To select appropriate min-points and eps for the clustering models, the distances to the N^{th} nearest neighbors were calculated and graphed. For both 5 and 10 min-points the slope of distance distribution increases exponentially where eps is approximately 400.

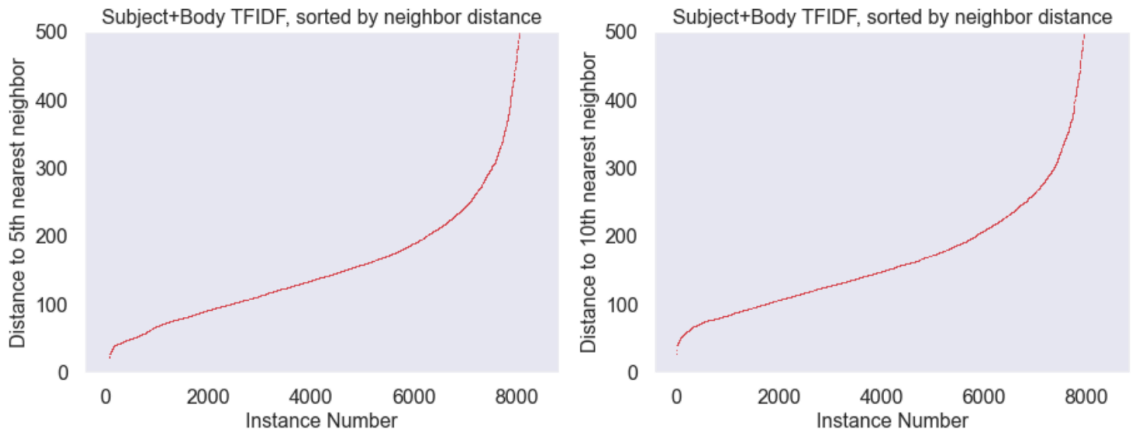

Figure 2: Determining eps from distance to nearest neighbor plot

For min-points = 5, eps=400, the number of clusters created was 28
For min-points = 10, eps=400, the number of clusters created was 6

Both clustering models were added as features for spam prediction models.

The DB-scan tool does not have internal capability to fit a model on a training set then predict clusters on unseen data. Assigning cluster numbers to the validation set by including it in a single clustering model combined with training data would require retraining the cluster model each time unseen data is received.

Instead after creating a clustering model with just the training data, a random forest model was created to predict the cluster number based on all other features available. Within the training set the random forest model predicted the cluster numbers with 100% accuracy. The random forest model was then used to make predictions on the cluster number for the validation set, allowing for spam predictions on unseen data without retraining cluster models.

Naïve-Bayes modeling

For prediction of spam classification we used the sklearn MultinomialNB package.

Grid searches were used to tune hyperparameters individually for models and in all cases the optimal parameters were:

alpha = 0.01 : Laplace smoothing options evaluated [0.01, 0.05, 0.1, 0.5, 1.0]
fit_prior = True : to learn class prior probabilities [False also evaluated]
class_prior = None : Prior probabilities of the classes adjusted according to the data. Evaluated [None, [.2,.8],[.25, .75]]

Separate models were created with the following features for comparison:

Model1: Subject only – includes the tf-idf using only the text of the subject field
Model2: Body only – includes the tf-idf using only the text of the body field
Model3 : Subject & Body – includes the individual tf-idf's for text of the subject and body
Model4 : Subject & Body & clusters– includes the individual tf-idf's for text of the subject and body as well as the cluster numbers for each of the two clustering models

The clustering features were one-hot encoded prior to naïve bayes modeling as these are categorical values rather than a linear relationship between cluster number and probability of being spam.

3 Results

The base model using vectorization of the subject field only performed well with 97% accuracy in predicting spam, however there is room for improvement in the precision and recall of the spam class.

Modeling with the vectorization of the body field improved the precision/recall on the spam class and improved accuracy to 99.4%

When the features from subject vectorization and body vectorization are both used, the accuracy improves slightly more to 99.5%.

Classification Performance Summary

	Subject tf-idf			Body tf-idf			Subject + Body tf-idf			Subject + Body tf-idf + clustering		
	precision	recall	f1-score	precision	recall	f1-score	precision	recall	f1-score	precision	recall	f1-score
Not_Spam	0.98	0.98	0.98	0.99	1.00	1.00	0.99	1.00	1.00	0.99	1.00	1.00
Spam	0.95	0.93	0.94	1.00	0.97	0.99	1.00	0.98	0.99	1.00	0.98	0.99
accuracy			0.970			0.994			0.995			0.995
macro avg	0.96	0.96	0.96	1.00	0.99	0.99	1.00	0.99	0.99	1.00	0.99	0.99
weighted avg	0.97	0.97	0.97	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99

Figure 3: Classification Performance metrics for validation set on each model

The confusion matrix shows that the 5 mis-classifications in the validation set were for emails that were truly spam but were predicted as not spam.

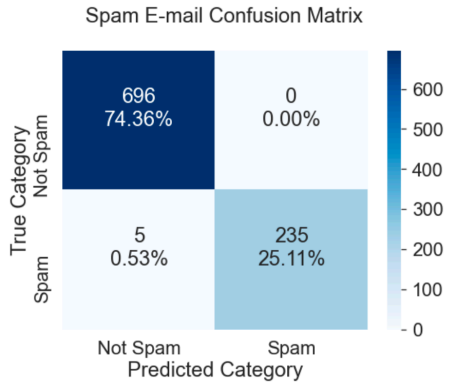

Figure 4: Confusion Matrix for final model

Examining the predicted probabilities for spam, shows large separation between classes and shifting the decision making boundary significantly lower than 0.5 does not improve precision/recall on spam without trade-offs to correctly predicting not-spam

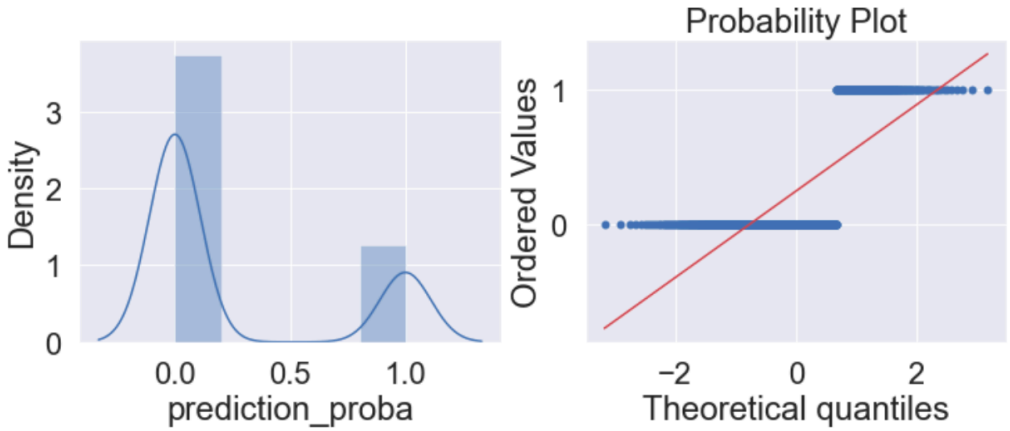

Figure 5: Prediction Probability distributions

Examining the 5 misclassifications show that all are from the “spam_2” subset rather than the “spam” set. We do not have more information on the difference in designating these two subsets, but it is possible that the spam_2 subset is harder to correctly predict.

subject	body	Group	Is_Spam	final_prediction	prediction_proba
Major Stock Play	We attempted to deliver this message to you wi...	spam_2	1	0	3.360162e-233
The Ultimate in PC Security and Surveillance.	PLEASE VISIT WWW.NOK-NOK.NET Nok-Nok :::: The...	spam_2	1	0	0.000000e+00
Re: change of plans	Hello you two, I am so sorry Catherine for no...	spam_2	1	0	0.000000e+00
ISA Article on Embedded Real-Time Linux Automa...	Industrial LINUX News: The June issue of the...	spam_2	1	0	5.146084e-238
[SA] EM1- Aren't we accorded equal opportunity...	Aren't Asian Pacfic Americans (APAs) accorde...	spam_2	1	0	0.000000e+00

Figure 6: Summary of emails that were misclassified

Conclusions

Adding clustering features did not improve the model further, so our model of choice would be the subject+body model and forgo the time associated with creating cluster.

The final model correctly filters out 98% of the spam messages without filtering out any of the truly important emails.

The model does not require any re-training for unseen data and can quickly compute the predictions of incoming e-mail.

Appendix

I. Code

A rendered notebook containing code for this analysis can be accessed at: https://nbviewer.org/github/rickfontenot/QTW/blob/main/Case%20Study%203/case3_rick.ipynb