OEIT6, Data Analytics 2022
Lab 6: Classification: Text Analysis

Objective: Separating Spam From Ham

Nearly every email user has at some point encountered a "spam" email, which is an unsolicited message often advertising a product, containing links to malware, or attempting to scam the recipient. Roughly 80-90% of more than 100 billion emails sent each day are spam emails, most being sent from botnets of malware-infected computers. The remainder of emails are called "ham" emails.

As a result of the huge number of spam emails being sent across the Internet each day, most email providers offer a spam filter that automatically flags likely spam messages and separates them from the ham. Though these filters use a number of techniques (e.g. looking up the sender in a so-called "Blackhole List" that contains IP addresses of likely spammers), most rely heavily on the analysis of the contents of an email via text analytics.

In this homework problem, we will build and evaluate a spam filter using a publicly available dataset (Note email.csv file in lab folder) The "ham" messages in this dataset come from the inbox of former Enron Managing Director for Research Vincent Kaminski, one of the inboxes in the Enron Corpus. One source of spam messages in this dataset is the SpamAssassin corpus, which contains hand-labeled spam messages contributed by Internet users. The remaining spam was collected by Project Honey Pot, a project that collects spam messages and identifies spammers by publishing email address that humans would know not to contact but that bots might target with spam. The full dataset we will use was constructed as roughly a 75/25 mix of the ham and spam messages.

The dataset contains just two fields:

- **text**: The text of the email.
- **spam**: A binary variable indicating if the email was spam.

Task

<u>Problem 1.1 – Loading the Dataset</u>

Begin by loading the dataset emails.csv into a data frame called emails (don't open the file with liboffice; import into python directly to avoid errors). Remember to pass the stringsAsFactors=FALSE option when loading the data.

How many emails are in the dataset?

How many of the emails are spam?

Which word appears at the beginning of every email in the dataset? Respond as a lower-case word with punctuation removed.

Could a spam classifier potentially benefit from including the frequency of the word that appears in every email?

Yes -- the number of times the word appears might help us differentiate spam from ham.

No -- the word appears in every email so this variable would not help us differentiate spam from ham.

The nchar() function counts the number of characters in a piece of text. How many characters are in the longest email in the dataset (where longest is measured in terms of the maximum number of characters)?

Problem 2.1 -Preparing the Corpus

Follow the standard steps to build and pre-process the corpus:

1) Build a new corpus variable called corpus.

2) Using tm_map, convert the text to lowercase.

3) Using tm_map, remove all punctuation from the corpus.

4) Using tm_map, remove all English stopwords from the corpus.

5) Using tm_map, stem the words in the corpus.

6) Build a document term matrix from the corpus, called dtm.

If the code length(stopwords("english")) does not return 174 for you, then please run the line of code in stopwords.tex (given in lab folder) file, which will store the standard stop words in a variable called sw. When removing stop words, use tm_map(corpus, removeWords, sw) instead of tm_map(corpus, removeWords, stopwords("english")).

How many terms are in dtm?

To obtain a more reasonable number of terms, limit dtm to contain terms appearing in at least 5% of documents, and store this result as spdtm (don't overwrite dtm, because we will use it in a later step of this homework). How many terms are in spdtm?

Problem 3.1 – Building machine learning models

First, create a variable called "emailsSparse" from "spdtm" using the command "emailsSparse = as.data.frame(as.matrix(spdtm))" and ensure it has legal variable names with "names(emailsSparse) = make.names(names(emailsSparse))". Then, copy the dependent variable from the original data frame called "emails" to "emailsSparse" using the command "emailsSparse$spam = as.factor(emails$spam)".

Next, set the random seed to 123 and use the sample.split function to split emailsSparse 70/30 into a training set called "train" and a testing set called "test". Make sure to perform this step on emailsSparse instead of emails.

Using the training set, train the following two machine learning models. The models should predict the dependent variable "spam", using all other available variables as independent variables. Please be patient, as these models may take a few minutes to train.

1) A CART model called spamCART, using the default parameters to train the model (don't worry about adding minbucket or cp or specifying losses for false positives and false negatives). Remember to add the argument method="class" since this is a binary classification problem.

2) A random forest model called spamRF, using the default parameters to train the model (don't worry about specifying mtry or ntree or nodesize or the losses for false positives and false negatives). Directly before training the random forest model, set the random seed to 123 (even though we've already done this earlier in the problem, it's important to set the seed right before training the model so we all obtain the same results. Keep in mind though that on certain operating systems, your results might still be slightly different).

Similar to logistic regression, CART and random forest can be told to give you predicted probabilities for classification problems. CART does this by returning the proportion of observations in the bucket of interest that fall into the relevant category, and random forest does this by returning the proportion of the trees that predict the relevant category. The following commands can be used to obtain predicted probabilities for the two fitted models on the training set:

predTrainCART = predict(spamCART)[,2]

predTrainRF = predict(spamRF, type="prob")[,2]

What is the training set accuracy of spamCART, using a threshold of 0.5 for predictions?

What is the training set accuracy of spamRF, using a threshold of 0.5 for predictions? (Remember that your answer might not match ours exactly, due to random behavior in the random forest algorithm on different operating systems.)

How many of the word stems "enron", "hou", "vinc", and "kaminski" appear in the CART tree?

What is the training set AUC of spamCART?

What is the training set AUC of spamRF?

Problem 4.1 – Evaluating on the Test Set

Obtain predicted probabilities for the testing set for each of the models, again ensuring that probabilities instead of classes are obtained. This can be achieved with the following code:

```
predTestCART = predict(spamCART, newdata=test)[,2]
```

```
predTestRF = predict(spamRF, newdata=test, type="prob")[,2]
```

What is the testing set accuracy of spamCART, using a threshold of 0.5 for predictions?

What is the testing set AUC of spamCART?

What is the testing set accuracy of spamRF, using a threshold of 0.5 for predictions?

What is the testing set AUC of spamRF?

Which model had the best testing set performance, in terms of accuracy and AUC?

-CART or Random Forest

Compare three models and show result

Deliverable:

You need to write document showing code that fulfills the objective requirement. You must write resulting inference for all intermediate steps, Must create github account and store your experiment write up there prior to next experiment.