The Analytics Edge

Test Your Knowledge of Text Analytics

1. 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 problem, you will build and evaluate a spam filter using a publicly available dataset first described in the 2006 conference paper "Spam Filtering with Naive Bayes – Which Naive Bayes?" by V. Metsis, I. Androutsopoulos, and G. Paliouras. 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, 1 indicating if the email was spam and 0 otherwise

   (a) Begin by loading the dataset **emails.csv** into a data frame called **emails**. 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?

   (b) Which word appears at the beginning of every email in the dataset?

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

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

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

(d) 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)?

(e) Which row contains the shortest email in the dataset? (Just like in the previous problem, shortest is measured in terms of the fewest number of characters). Write down the corresponding email.

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

- Load the tm package.
- Build a new corpus variable called corpus.
- Using tm_map, convert the text to lowercase.
- Using tm_map, remove all punctuation from the corpus.
- Using tm_map, remove all English stopwords from the corpus.
- Using tm_map, stem the words in the corpus.
- Build a document term matrix from the corpus, called dtm

How many terms are in dtm?

(g) 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 later). How many terms are in spdtm?

(h) Build a data frame called emailsSparse from spdtm, and use the make.names function to make the variable names of emailsSparse valid. colSums() is an R function that returns the sum of values for each variable in our data frame. Our data frame contains the number of times each word stem (columns) appeared in each email (rows). Therefore, colSums(emailsSparse) returns the number of times a word stem appeared across all the emails in the dataset. What is the word stem that shows up most frequently across all the emails in the dataset?

(i) Add a variable called "spam" to emailsSparse containing the email spam labels. How many word stems appear at least 5000 times in the ham emails in the dataset? Which word stems are these?

(j) How many word stems appear at least 1000 times in the spam emails in the dataset? Which word stems are these?

(k) The lists of most common words are significantly different between the spam and ham emails. What does this likely imply?

i. The frequencies of these most common words are unlikely to help differentiate between spam and ham.

ii. The frequencies of these most common words are likely to help differentiate between spam and ham.

(l) Several of the most common word stems from the ham documents, such as "enron", "hou" (short for Houston), "vinc" (the word stem of "Vince") and "kaminski", are likely specific to Vincent Kaminski's inbox. What does this mean about the applicability of the text analytics models we will train for the spam filtering problem?

i. The models we build are still very general, and are likely to perform well as a spam filter for nearly any other person.

ii. The models we build are personalized, and would need to be further tested before being used as a spam filter for another person.

(m) First, convert the dependent variable to a factor with
> emailsSparse$spam <− as.factor(emailsSparse$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 three 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.

- A logistic regression model called spamLog. You may see a warning message here - we'll discuss this more later.
- A CART model called spamCART, using the default parameters to train the model. Directly before training the CART model, set the random seed to 123.
- A random forest model called spamRF, using the default parameters to train the model. 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).

For each model, obtain the predicted spam probabilities for the training set.
You may have noticed that training the logistic regression model yielded the messages "algorithm did not converge" and "fitted probabilities numerically 0 or 1 occurred". Both of these messages often indicate overfitting and in some case corresponds to severe overfitting, often to the point that the training set observations are fit perfectly by the model. Let's investigate the predicted probabilities from the logistic regression model.
How many of the training set predicted probabilities from spamLog are less than 0.00001?
How many of the training set predicted probabilities from spamLog are more than 0.99999?
How many of the training set predicted probabilities from spamLog are between 0.00001 and 0.99999?

(n) How many variables are labeled as significant (at the p=0.05 level) in the logistic regression summary output?

(o) How many of the word stems "enron", "hou", "vinc", and "kaminski" appear in the CART tree? Recall that we suspect these word stems are specific to Vincent Kaminski and might affect the generalizability of a spam filter built with his ham data.

(p) What is the training set accuracy of spamLog, using a threshold of 0.5 for predictions? What is the training set AUC of spamLog?

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

(r) What is the training set AUC of spamCART? (Remember that you have to pass the prediction function predicted probabilities.)

(s) What is the training set accuracy of spamRF, using a threshold of 0.5 for predictions? (Remember that your have to use type="prob" in your prediction for random forest.)

(t) What is the training set AUC of spamRF? (Remember to pass the argument type="prob" to the predict function to get predicted probabilities for a random forest model. The probabilities will be the second column of the output.)

(u) Which of the models have the best training set performance, in terms of accuracy and AUC?

- Logistic regression
- CART
- Random forest

(v) Obtain predicted probabilities for the testing set for each of the models, again ensuring that probabilities instead of classes are obtained. What is the testing set accuracy of spamLog, using a threshold of 0.5 for predictions?

(w) What is the testing set AUC of spamLog? 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?

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

- Logistic regression
- CART
- Random forest

(y) Which model demonstrated the greatest degree of overfitting?

- Logistic regression
- CART
- Random forest