MITx: 15.071x The Analytics Edge

SEPARATING SPAM FROM HAM (PART 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 homework problem, we 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 indicating if the email was spam.

IMPORTANT NOTE: This problem (Separating Spam from Ham) continues on the next page with additional exercises. The second page is optional, but if you want to try it out, remember to save your work so you can start the next page where you left off here.

PROBLEM 1.1 - LOADING THE DATASET (1/1 point)

Begin by loading the dataset emails.csv into a data frame called emails. Remember to pass the stringsAsFactors=FALSE option when loading the data.

EXPLANATION

You can load the dataset with:

emails = read.csv("emails.csv", stringsAsFactors=FALSE)

How many emails are in the dataset?

5728 ✔ **Answer:** 5728

EXPLANATION

The number of emails can be read from str(emails) or nrow(emails).

CHECK **SAVE** HIDE ANSWER *You have used 1 of 3 submissions*

PROBLEM 1.2 - LOADING THE DATASET (1/1 point)

How many of the emails are spam?

1368 ✔ **Answer:** 1368

EXPLANATION

This can be read from table(emails$spam).

CHECK **SAVE** HIDE ANSWER *You have used 1 of 3 submissions*

PROBLEM 1.3 - LOADING THE DATASET (1/1 point)

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

subject ✔ **Answer:** subject

EXPLANATION

You can review emails with, for instance, emails$text[1] or emails$text[1000]. Every email begins with the word "Subject:".

CHECK **SAVE** **HIDE ANSWER** *You have used 1 of 3 submissions*

PROBLEM 1.4 - LOADING THE DATASET (1 point possible)

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

- (•) No -- the word appears in every email so this variable would not help us differentiate spam from ham. ✖
- ( ) Yes -- the number of times the word appears might help us differentiate spam from ham. ✔

EXPLANATION

We know that each email has the word "subject" appear at least once, but the frequency with which it appears might help us differentiate spam from ham. For instance, a long email chain would have the word "subject" appear a number of times, and this higher frequency might be indicative of a ham message.

HIDE ANSWER *You have used 1 of 1 submissions*

PROBLEM 1.5 - LOADING THE DATASET (1/1 point)

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)?

43952 ✔ **Answer:** 43952

EXPLANATION

The maximum length can be obtained with max(nchar(emails$text)).

CHECK **SAVE** HIDE ANSWER *You have used 1 of 3 submissions*

PROBLEM 1.6 - LOADING THE DATASET (1/1 point)

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.)

1992 ✔

1992

Answer: 1992

EXPLANATION

The minimum length, 13 characters, can be determined with min(nchar(emails$text)). We can see that this is achieved only in email 1992 from which(nchar(emails$text) == 13). An easier approach would be which.min(nchar(emails$text)).

CHECK **SAVE** HIDE ANSWER *You have used 1 of 3 submissions*

PROBLEM 2.1 - PREPARING THE CORPUS (2/2 points)

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 this 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?

28687 ✔ **Answer:** 28687

EXPLANATION

These steps can be accomplished by running:

```
corpus = Corpus(VectorSource(emails$text))

corpus = tm_map(corpus, tolower)

corpus = tm_map(corpus, PlainTextDocument)

corpus = tm_map(corpus, removePunctuation)

corpus = tm_map(corpus, removeWords, stopwords("english"))

corpus = tm_map(corpus, stemDocument)

dtm = DocumentTermMatrix(corpus)

dtm
```

From the dtm summary output, we can read that it contains 28687 terms.

CHECK **SAVE** **HIDE ANSWER** *You have used 1 of 5 submissions*

PROBLEM 2.2 - PREPARING THE CORPUS (1/1 point)

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?

330 ✔ **Answer:** 330

EXPLANATION

This can be accomplished with:

spdtm = removeSparseTerms(dtm, 0.95)

spdtm

From the spdtm summary output, it contains 330 terms.

CHECK **SAVE** **HIDE ANSWER** *You have used 1 of 3 submissions*

PROBLEM 2.3 - PREPARING THE CORPUS (2/2 points)

Build a data frame called emailsSparse from spdtm, and use the make.names function to make the variable names of emailsSparse valid.

EXPLANATION

This can be accomplished with:

emailsSparse = as.data.frame(as.matrix(spdtm))

colnames(emailsSparse) = make.names(colnames(emailsSparse))

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? Hint: think about how you can use sort() or which.max() to pick out the maximum frequency.

enron ✔ **Answer:** enron

EXPLANATION

colSums(emailsSparse) contains the sum of all the values for each column in our data frame. Since the values in the data frame are the frequencies of the stem in the column for the email in the row, these column sums represent the frequencies of the stems across all emails.

We can either use sort() or which.max() to pick out the most common word:

sort(colSums(emailsSparse))

which.max(colSums(emailsSparse))

CHECK **SAVE** **HIDE ANSWER** *You have used 1 of 5 submissions*

PROBLEM 2.4 - PREPARING THE CORPUS (1/1 point)

Add a variable called "spam" to emailsSparse containing the email spam labels. You can do this by copying over the "spam" variable from the original data frame (remember how we did this in the Twitter lecture).

EXPLANATION

This can be accomplished with:

emailsSparse$spam = emails$spam

How many word stems appear at least 5000 times in the ham emails in the dataset? Hint: in this and the next question, remember not to count the dependent variable we just added.

6 ✔

6

Answer: 6

EXPLANATION

We can read the most frequent terms in the ham dataset with sort(colSums(subset(emailsSparse, spam == 0))). "enron", "ect", "subject", "vinc", "will", and "hou" appear at least 5000 times in the ham dataset.

FINAL CHECK | SAVE | HIDE ANSWER *You have used 2 of 3 submissions*

PROBLEM 2.5 - PREPARING THE CORPUS (1/1 point)

How many word stems appear at least 1000 times in the spam emails in the dataset?

3 ✔ **Answer:** 3

EXPLANATION

We can limit the dataset to the spam emails with subset(emailsSparse, spam == 1). Therefore, we can read the most frequent terms with sort(colSums(subset(emailsSparse, spam == 1))). "subject", "will", and "compani" are the three stems that appear at least 1000 times. Note that the variable "spam" is the dependent variable and is not the frequency of a word stem.

FINAL CHECK | SAVE | HIDE ANSWER *You have used 2 of 3 submissions*

PROBLEM 2.6 - PREPARING THE CORPUS (1/1 point)

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

- ○ The frequencies of these most common words are unlikely to help differentiate between spam and ham.
- ◉ The frequencies of these most common words are likely to help differentiate between spam and ham. ✔

EXPLANATION

A word stem like "enron", which is extremely common in the ham emails but does not occur in any spam message, will help us correctly identify a large number of ham messages.

HIDE ANSWER *You have used 1 of 1 submissions*

PROBLEM 2.7 - PREPARING THE CORPUS (1/1 point)

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?

- ○ The models we build are still very general, and are likely to perform well as a spam filter for nearly any other person.
- ◉ The models we build are personalized, and would need to be further tested before being used as a spam filter for another person. ✔

EXPLANATION

The ham dataset is certainly personalized to Vincent Kaminski, and therefore it might not generalize well to a general email user. Caution is definitely necessary before applying the filters derived in this problem to other email users.

HIDE ANSWER *You have used 1 of 1 submissions*

PROBLEM 3.1 - BUILDING MACHINE LEARNING MODELS (3/3 points)

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.

EXPLANATION

These steps can be accomplished with:

```
emailsSparse$spam = as.factor(emailsSparse$spam)

set.seed(123)

library(caTools)

spl = sample.split(emailsSparse$spam, 0.7)

train = subset(emailsSparse, spl == TRUE)

test = subset(emailsSparse, spl == FALSE)
```

Using the training set, train the following three 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 logistic regression model called spamLog. You may see a warning message here - we'll discuss this more later.

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

3) A random forest model called spamRF, using the default parameters to train the model (don't worry about specifying ntree or nodesize). 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).

EXPLANATION

These models can be trained with the following code:

```
spamLog = glm(spam~., data=train, family="binomial")

spamCART = rpart(spam~., data=train, method="class")

set.seed(123)

spamRF = randomForest(spam~., data=train)
```

For each model, obtain the predicted spam probabilities for the **training set**. Be careful to obtain probabilities instead of predicted classes, because we will be using these values to compute training set AUC values. Recall that you can obtain probabilities for CART models by not passing any type parameter to the predict() function, and you can obtain probabilities from a random forest by adding the argument type="prob". For CART and random forest, you need to select the second column of the output of the predict() function, corresponding to the probability of a message being spam.

EXPLANATION

These probabilities can be obtained with:

```
predTrainLog = predict(spamLog, type="response")

predTrainCART = predict(spamCART)[,2]

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

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 the first indicates particularly 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?

3046 ✔ **Answer:** 3046

How many of the training set predicted probabilities from spamLog are more than 0.99999?

954 ✔ **Answer:** 954

How many of the training set predicted probabilities from spamLog are between 0.00001 and 0.99999?

10 ✔ **Answer:** 10

EXPLANATION

To check the number of probabilities with these characteristics, we can use:

table(predTrainLog < 0.00001)

table(predTrainLog > 0.99999)

table(predTrainLog >= 0.00001 & predTrainLog <= 0.99999)

You might have gotten slightly different answers than the ones you see here, because the glm function has a hard time converging with this many independent variables. That's okay - your answers should still be marked as correct.

CHECK **SAVE** **HIDE ANSWER** *You have used 1 of 5 submissions*

PROBLEM 3.2 - BUILDING MACHINE LEARNING MODELS (1/1 point)

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

0 ✔ **Answer:** 0

EXPLANATION

From summary(spamLog), we see that none of the variables are labeled as significant (a symptom of the logistic regression algorithm not converging).

CHECK **SAVE** **HIDE ANSWER** *You have used 1 of 3 submissions*

PROBLEM 3.3 - BUILDING MACHINE LEARNING MODELS (1/1 point)

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.

2 ✔ **Answer:** 2

EXPLANATION

From prp(spamCART), we see that "vinc" and "enron" appear in the CART tree as the top two branches, but that "hou" and "kaminski" do not appear.

FINAL CHECK | **SAVE** | **HIDE ANSWER** *You have used 1 of 2 submissions*

PROBLEM 3.4 - BUILDING MACHINE LEARNING MODELS (1/1 point)

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

0.9990025 ✔

0.9990025

Answer: 0.9990025

EXPLANATION

This can be obtained with:

table(train$spam, predTrainLog > 0.5)

The accuracy is (3052+954)/nrow(train).

CHECK | **SAVE** | **HIDE ANSWER** *You have used 1 of 3 submissions*

PROBLEM 3.5 - BUILDING MACHINE LEARNING MODELS (1/1 point)

What is the training set AUC of spamLog?

0.9999959 ✔ **Answer:** 0.9999959

EXPLANATION

This can be obtained with:

predictionTrainLog = prediction(predTrainLog, train$spam)

as.numeric(performance(predictionTrainLog, "auc")@y.values)

CHECK SAVE HIDE ANSWER *You have used 1 of 3 submissions*

PROBLEM 3.6 - BUILDING MACHINE LEARNING MODELS (1/1 point)

What is the training set accuracy of spamCART, using a threshold of 0.5 for predictions? (Remember that if you used the type="class" argument when making predictions, you automatically used a threshold of 0.5. If you did not add in the type argument to the predict function, the probabilities are in the second column of the predict output.)

0.942394 ✔

0.942394

Answer: 0.942394

EXPLANATION

This can be obtained with:

table(train$spam, predTrainCART > 0.5)

Then the accuracy is (2885+894)/nrow(train)

CHECK SAVE HIDE ANSWER *You have used 1 of 5 submissions*

PROBLEM 3.7 - BUILDING MACHINE LEARNING MODELS (1/1 point)

What is the training set AUC of spamCART? (Remember that you have to pass the prediction function predicted probabilities, so don't include the type argument when making predictions for your CART model.)

0.9696044 ✔ **Answer:** 0.9696044

EXPLANATION

This can be obtained with:

predictionTrainCART = prediction(predTrainCART, train$spam)

as.numeric(performance(predictionTrainCART, "auc")@y.values)

CHECK **SAVE** HIDE ANSWER *You have used 1 of 5 submissions*

PROBLEM 3.8 - BUILDING MACHINE LEARNING MODELS (1/1 point)

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.)

0.9793017 ✔

0.9793017

Answer: 0.9793017

EXPLANATION

This can be obtained with:

table(train$spam, predTrainRF > 0.5)

And then the accuracy is (3013+914)/nrow(train)

CHECK **SAVE** HIDE ANSWER *You have used 2 of 5 submissions*

PROBLEM 3.9 - BUILDING MACHINE LEARNING MODELS (2/2 points)

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.)

0.9979116 ✔ **Answer:** 0.9979116

EXPLANATION

This can be obtained with:

predictionTrainRF = prediction(predTrainRF, train$spam)

as.numeric(performance(predictionTrainRF, "auc")@y.values)

CHECK **SAVE** HIDE ANSWER *You have used 1 of 5 submissions*

PROBLEM 3.10 - BUILDING MACHINE LEARNING MODELS (1/1 point)

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

- (•) Logistic regression ✔
- ( ) CART
- ( ) Random forest

EXPLANATION

In terms of both accuracy and AUC, logistic regression is nearly perfect and outperforms the other two models.

HIDE ANSWER *You have used 1 of 1 submissions*

PROBLEM 4.1 - EVALUATING ON THE TEST SET (1/1 point)

Obtain predicted probabilities for the testing set for each of the models, again ensuring that probabilities instead of classes are obtained.

EXPLANATION

The predicted probabilities can be obtained with:

predTestLog = predict(spamLog, newdata=test, type="response")

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

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

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

0.9505239 ✔

0.9505239

Answer: 0.9505239

EXPLANATION

This can be obtained with:

table(test$spam, predTestLog > 0.5)

Then the accuracy is (1257+376)/nrow(test)

CHECK **SAVE** HIDE ANSWER *You have used 1 of 3 submissions*

PROBLEM 4.2 - EVALUATING ON THE TEST SET (1/1 point)

What is the testing set AUC of spamLog?

0.9627 ✔ **Answer:** 0.9627517

EXPLANATION

This can be obtained with:

predictionTestLog = prediction(predTestLog, test$spam)

as.numeric(performance(predictionTestLog, "auc")@y.values)

CHECK **SAVE** HIDE ANSWER *You have used 1 of 3 submissions*

PROBLEM 4.3 - EVALUATING ON THE TEST SET (1/1 point)

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

0.9394 ✔

0.9394

Answer: 0.9394645

EXPLANATION

This can be obtained with:

table(test$spam, predTestCART > 0.5)

Then the accuracy is (1228+386)/nrow(test)

CHECK **SAVE** HIDE ANSWER *You have used 1 of 3 submissions*

PROBLEM 4.4 - EVALUATING ON THE TEST SET (1/1 point)

What is the testing set AUC of spamCART?

0.9631 ✔ **Answer:** 0.963176

EXPLANATION

This can be obtained with:

predictionTestCART = prediction(predTestCART, test$spam)

as.numeric(performance(predictionTestCART, "auc")@y.values)

CHECK **SAVE** HIDE ANSWER *You have used 1 of 3 submissions*

PROBLEM 4.5 - EVALUATING ON THE TEST SET (1/1 point)

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

0.9755 ✔

0.9755

Answer: 0.975553

EXPLANATION

This can be obtained with:

table(test$spam, predTestRF > 0.5)

Then the accuracy is (1290+385)/nrow(test)

CHECK **SAVE** HIDE ANSWER *You have used 1 of 3 submissions*

PROBLEM 4.6 - EVALUATING ON THE TEST SET (1/1 point)

What is the testing set AUC of spamRF?

0.9975 ✔ **Answer:** 0.9975656

EXPLANATION

This can be obtained with:

predictionTestRF = prediction(predTestRF, test$spam)

as.numeric(performance(predictionTestRF, "auc")@y.values)

CHECK **SAVE** **HIDE ANSWER** *You have used 1 of 3 submissions*

PROBLEM 4.7 - EVALUATING ON THE TEST SET (1/1 point)

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

- ○ Logistic regression
- ○ CART
- ◉ Random forest ✔

EXPLANATION

The random forest outperformed logistic regression and CART in both measures, obtaining an impressive AUC of 0.997 on the test set.

HIDE ANSWER *You have used 1 of 1 submissions*

PROBLEM 4.8 - EVALUATING ON THE TEST SET (1/1 point)

Which model demonstrated the greatest degree of overfitting?

- (•) Logistic regression ✔
- ( ) CART
- ( ) Random forest

EXPLANATION

Both CART and random forest had very similar accuracies on the training and testing sets. However, logistic regression obtained nearly perfect accuracy and AUC on the training set and had far-from-perfect performance on the testing set. This is an indicator of overfitting.

HIDE ANSWER *You have used 1 of 1 submissions*

IMPORTANT NOTE: The second part of this homework assignment is optional, and is on the next page. If you want to complete the optional assignment, remember to save your work so you can start the next page where you left off here.

Please remember not to ask for or post complete answers to homework questions in this discussion forum.

Show Discussion New Post