Cross Validation and the $k^{th}$-Nearest Neighbor Classifier of SPAM

DUE: Oct 27

In this project, we focus on the problem of classifying email messages as spam or ham. We will consider two aspects of this problem: nearest neighbor based classification methods and statistical cross-validation. The data we use are derived from the mail messages from the Spam Assassin website http://spamassassin.apache.org/. It is available as an R data frame, with one record for each email and one column for each derived variable. In fact there are two data frames, the one called derivedEmails will be used to determine the classification method and other called NewEmails will be used to test the classifier developed on the derivedEmails data. The data frames and a description of the variables are available on the class website.

Our goal is to create a classifier to predict whether a new mail message is spam or ham. We will use what is called a $k^{th}$-nearest neighbor classifier, which finds the $k$ emails from a training set that are closest to the new email and classifies the email as ham or spam according to whether the majority of the neighboring emails are ham or spam. That is, the $k$ nearest neighbors to the new email message vote for the message being ham or spam, according to whether they are ham or spam. The majority wins. (What should be done if there is a tie?)

To find the $k$ nearest neighbors to an email message, we need to be able to compute the distance between any pair of emails. How do we do that? This is where the variables come into play. For each email in the training set we find the distance between the value of the variables for the email in the training set and the new email. To compute distances, we need a metric. One well known metric is Euclidean distance, which may be appropriate for continuous valued variables, but not for logical variables. (Why?) A binary metric may be more appropriate for variables that take on one of two possible values. The R function `dist` computes such distances for various metrics.

To determine the value of $k$, we want to compare the performance of the $k^{th}$-nearest neighbor classifier for various values of $k$. At one extreme, we have nearest neighbor classification, which takes $k$ to be 1. That is, the new email is classified as spam or ham depending on whether its nearest neighbor is spam or ham, respectively. At the other extreme, we could use all the data in the training set to classify the new email. In this case, the classification would be
the same for any email, i.e. it would be ham if the majority of the emails in the training set are ham.

To choose a value for $k$, we compare how well the method does when we know the truth, i.e. when we know whether the new email is spam or ham. We can compute the probability that a mistake is made. There are two types of mistakes: a Type I error is when an actual ham message is classified as spam, and a Type II error occurs when a spam message is misclassified as ham. We use cross-validation to determine the value of $k$ so that we can control these errors.

The following are your tasks:

1. Working with the derivedEmails data frame, determine an appropriate metric to find the distance between the emails.

The first issue you face is how to compute the distance between any two emails. For continuous variables, such as the number of lines in the body of the email, the percentage of capital letters in the subject of the email, and the number of recipients, Euclidean and Manhattan metrics may be appropriate. For these three variables, numLinesInBody, percentCapitals, and numRecipients, we would compute the Euclidean distance between emails $i$ and $j$ as follows:

$$
\sqrt{((\text{numLinesInBody}_i - \text{numLinesInBody}_j)^2 + (\text{percentCapitals}_i - \text{percentCapitals}_j)^2 + (\text{numRecipients}_i - \text{numRecipients}_j)^2)}
$$

and the Manhattan distance between them is:

$$
|\text{numLinesInBody}_i - \text{numLinesInBody}_j| + |\text{percentCapitals}_i - \text{percentCapitals}_j| + |\text{numRecipients}_i - \text{numRecipients}_j|
$$

But the percentage of capitals in the subject will always be between 0 and 1, whereas the number of lines in the body of an email can get quite large. Should you normalize the continuous variables before computing the distance between two records? One way to do this is center the variable on the median value and to scale it by the median absolute deviation, i.e.

$$
\frac{\text{numLinesInBody}_i - \text{median}(\text{numLinesInBody})}{\text{mad}(\text{numLinesInBody})}.
$$

Alternatively, the mean and mean absolute deviation could standardize a variable. Other metrics to consider are the maximum (sup) norm and the Canberra distance, which includes a standardization.
For logical variables, even if they are converted to 0-1 values, the Euclidean norm seems inappropriate to measure distance. Instead, either the asymmetric binary or the symmetric binary metrics are more appropriate. The asymmetric distance between two records with variables, say isRe, replyUnderline, and multipartText, would be 0, 1/3, 2/3, or 1 according to the following rule: 1 − the proportion of variables that are 1 for both records among those variables that have a 1 for at least one record. When neither record has any 1s, the distance is 0.

The dist function and the daisy function compute distances between rows of a matrix or data frame, where the columns represent variables. You might need to subdivide your data frame into continuous and logical parts to more easily handle these different data types. If you do this, you need to consider how to recombine the two distances.

2. Write an R function that uses \textit{v}-fold cross-validation on the derivedEmails data frame to compare the predictive ability of the nearest neighbor method for various values of \( k \). In addition, you may consider exploring the behavior of choosing different values for \( v \) in \( v \)-fold cross validation. R has several functions related to \( k \)th nearest neighbor in the \textit{class} package, but they use euclidean distance only. Also, some of the functions compute the \( k \)th nearest neighbor classification for one \( k \) at a time, but the distance between two observations only needs to be computed once to figure out the \( k \)th nearest neighbor for values of \( k \) from 1 to the size of the training set.

3. Consider alternative metrics, and use cross-validation to choose both the best \( k \) and metric. Plot the Type I and Type II error rates for the different values of \( k \) and different metrics. Select a value for \( k \) and a metric that defines a classifier with a "suitable" error rate.

The Type I error rate is the rate at which ham messages are miscategorized as spam, and the Type II error rate is the rate at which the spam is misclassified as ham. For example, of the 1475 spam messages, if 735 are classified as ham and 740 are classified as spam then the Type II error rate for this set of data is 735/1475 = 0.498. The logical vector that contains the indicator for spam and a logical that contains an indicator for whether the classification is correct or not can be used to find these two types of errors.

Be sure to plot these two error rates (on the same canvas) for various values of \( k \) and various metrics.

4. How well does your choice of \( k \) do in classifying the emails in the new set called NewSet? Here you want to use the entire set of emails in the derivedEmails data frame as the training set and NewSet as a validation set. Explain your findings using a graphical analyses. To do this, you might want to first examine the prediction made by the classification tree
method, as it can give you an idea about which variables may be important.

5. Compare your optimal nearest neighbor procedure with the method of classification trees available in the \texttt{rpart} library. Use derivedEmails to settle on a tree (cross-validation is built into the \texttt{rpart} procedure) and then evaluate the tree’s predictive ability on NewSet. Does the classification tree approach outperform the nearest neighbor approach?

A classification tree is an intuitively simple classification method. Beginning at the root node of the tree, the data split or branch into two groups according to the value of a variable in the data frame. For example, the split may be according to the value of \texttt{percentCapitals}, where the data are divided into two groups according to whether the value of \texttt{percentCapitals} is above 0.11 or not. Subsequent splits are made along the resulting two branches in a similar fashion, that is, each split considers the value of a single variable and subdivides the subset of data at that node into two subgroups. The variables must all be either factors or numeric. The splits are made to make the resulting subgroups as similar as possible, i.e. to reduce the prediction error as much as possible.

The \texttt{rpart} returns an object of class \texttt{rpart}. This means that many functions know how to handle this object. For example, if you use the \texttt{plot} with an \texttt{rpart} object, it will plot the subdivisions of the data as a tree. Also, the \texttt{predict} function (see help on \texttt{predict.rpart}) allows you to classify records using an \texttt{rpart} object. To find out more about the \texttt{rpart} library, read the documentation at \url{http://cran.r-project.org/doc/packages/rpart.pdf}, or other documentation that you may find in a Google search.

6. Write up your findings in a 5-10 page paper, including plots and tables. The paper must be submitted in pdf. In addition, submit your R code in a plain text file.

NOTE: You are to work in groups of 2-3 for this project. Only one report and set of code needs to be turned in. Be sure to include in the report the name and SID number of each group member.