In this assignment, you will be working with data from the SeaFlow environmental flow cytometry instrument.

A flow cytometer delivers a flow of particles through capilliary. By shining lasers of different wavelengths and measuring the absorption and refraction patterns, you can determine how large the particle is and some information about its color and other properties, allowing you to detect it.

The technology was developed for medical applciations, where the particles were potential pathogens in, say, serum, and the goal was to give a diagnosis. But the technology was adapted for use in environmental science to understand microbial population profiles.

The SeaFlow instrument, developed by the Armbrust Lab at the University of Washington, is unique in that it is deployed on research vessels and takes continuous measurements of population profiles in the open ocean.

The scale of the data can be quite large, and is expected to grow significantly: A two-week cruise from one vessel can generate hundreds of gigabytes per day, and the vision is to deploy one of these instruments on not only research vessels but the commercial shipping fleet as well.

While there are a number of challenging analytics tasks associated with this data, a central task is classification of particles. Based on the optical measurements of the particle, it can be identified as one of several populations.

Configuring R

A useful thread in the discussion forum points out two important items to get R configured properly on the class virtual machine:

1. The need to run Rscript setup.r as sudo Rscript setup.r to avoid permissions problems

2. The need to setup ~/.Rprofile to contain the following, so that a CRAN mirror will be selected:

```
local({r <- getOption("repos");         r["CRAN"] <- "http://cran.r-project.org"
  ; options(repos=r)})
```

Dataset

You are provided a dataset that represents a 21 minute sample from the vessel in a file seaflow_21min.csv. This sample has been pre-processed to remove the calibration "beads" that are passed through the system for monitoring, as well as some other particle types.

The columns of this dataset are as follows:

```
file_id, time, cell_id, d1, d2, fsc_small, fsc_perp, fsc_big, pe, chl_small,
  chl_big, pop
```

· file_id: The data arrives in files, where each file represents a three-minute window; this field represents which file the data came from. The number is ordered by time, but is otherwise not significant.

· time: This is an integer representing the time the particle passed through the instrument. Many particles may arrive at the same time; time is not a key for this relation.

· cell_id: A unique identifier for each cell WITHIN a file. (file_id, cell_id) is a key for this relation.

· d1, d2: Intensity of light at the two main sensors, oriented perpendicularly. These sensors are primarily used to determine whether the particles are properly centered in the stream. Used primarily in preprocesssing; they are unlikely to be useful for classification.

· fsc_small, fsc_perp, fsc_big: Forward scatter small, perpendicular, and big. These values help distingish different sizes of particles.

· pe: A measurement of phycoerythrin fluorescence, which is related to the wavelength associated with an orange color in microorganisms

· chl_small, chl_big: Measurements related to the wavelength of light corresponding to chlorophyll.

· pop: This is the class label assigned by the clustering mechanism used in the production system. It can be considered "ground truth" for the purposes of the assignment, but note that there are particles that cannot be unambiguously classified, so you should not aim for 100% accuracy. The values in this column are crypto, nano, pico, synecho, and ultra

Overview

You will work with the dataset in R, using and evaluating a few different classification methods. You will not turn in the code itself; you will answer questions about the data and methods that (hopefully) demonstrate that you have written correct code. As usual, we will tend not to provide specific syntax; you will look up the appropriate documentation and experiment with the language to elicit the correct answer. That said, learning R as a programming language is not a goal of this assignment (or this course) -- there are plenty of courses that aim to teach the use of R. The goal of this assignment is to use R to experiment with a real dataset -- including its idiosyncrasies -- and to consider the strengths and weaknesses of a few popular methods for supervised learning.

If you wish to follow a tutorial in R, there are many to choose from, but you might try this one.

First thing you should do, as always, is do a git pull to fetch the latest version of the course repository. The file seaflow_21min.csv is in the repository.

To install the necessary packages in your system, you can run the script setup.r as follows:

```
$ Rscript setup.r
```

Alternatively, you can just install the packages yourself; open the setup script to see which ones you need.

Step 1: Read and summarize the data

Using R, read the file seaflow_21min.csv and get the overall counts for each category of particle. You may consider using the functions read.csv and summary.

Answer Questions 1 and 2

Step 2: Split the data into test and training sets

Divide the data into two equal subsets, one for training and one for testing. Make sure to divide the data in an unbiased manner.

You might consider using either the createDataPartition function or the sample function, although there are many ways to achieve the goal.

Answer Question 3

Step 3: Plot the data

Plot pe against chl_small and color by pop

I recommend using the function ggplot in the library ggpplot2 rather than using base R functions, but this is not required.

Answer Question 4

Step 4: Train a decision tree.

Install the rpart library if you do not have it, and load the library.

Many statistical models in R provide an interface of the form

```
model <- train(formula, dataframe)
```

You can then use the model object to make predictions by passing it to the predict function.

A formula object describes the relationship between the independent variables and the response variable, and can be expressed with the syntax

```
response ~ var1 + var2 + var3
```

and used with the formula function to construct the formula object itself:

```
fol <- formula(response ~ var1 + var2 + var3)
```

The rpart library uses this convention. Assuming your training data is in a data frame called training and you have constructed a formula objec tcalled fol, you can construct a decision tree using the following syntax (included here to avoid you struggling with a couple of unusual aspects of R):

```
model <- rpart(fol, method="class", data=training)
```

Train a tree as a function of the sensor measurements: fsc_small + fsc_perp + fsc_big + pe + chl_big + chl_small

Print the model object using the print function print(model)

The output is a set of decision nodes, one node per line. Each line is indented indicating the height of the tree. Here is a bogus example of a tree:

1) root 33456 22345 nano (0.0016 0.17 0.29 0.25 0.28) 2) chl_small< 31000 26238 15772 pico (0 0.22 0.4 3.8e-05 0.38) 4) fsc_perp< 2040 11430 1913 pico (0 8.7e-05 0.83 8.7e-05 0.17) * 10) chl_small>=12000 7065 628 nano (0 0.88 0 0 0.12) * 11) chl_small< 12000 9000 2232 ultra (0 0.13 0.097 0 0.77) * 5) fsc_perp>=2040 14808 5500 ultra (0 0.39 0.064 0 0.55) 3) chl_small>=31000 9933 780 synecho (0.0058 0.054 0.0044 0.92 0.014) 6) pe>=17532 681 156 nano (0.085 0.77 0 0.069 0.075) * 7) pe< 17532 9252 146 synecho (0 0.0014 0.0048 0.98 0.0096) *

```
1) root 33456 22345 nano (0.0016 0.17 0.29 0.25 0.28)
  2) chl_small< 31000 26238 15772 pico (0 0.22 0.4 3.8e-05 0.38)
    4) fsc_perp< 2040 11430 1913 pico (0 8.7e-05 0.83 8.7e-05 0.17)  *
      10) chl_small>=12000 7065 628 nano (0 0.88 0 0 0.12) *
      11) chl_small< 12000 9000 2232 ultra (0 0.13 0.097 0 0.77) *
    5) fsc_perp>=2040 14808  5500 ultra (0 0.39 0.064 0 0.55)
  3) chl_small>=31000 9933 780 synecho (0.0058 0.054 0.0044 0.92 0.014)
    6) pe>=17532 681 156 nano (0.085 0.77 0 0.069 0.075) *
    7) pe< 17532 9252 146 synecho (0 0.0014 0.0048 0.98 0.0096) *
```

To make a prediction, walk down the tree applying the predicates to determine which branch to take. For example, in this bogus tree, a particle with chl_small=25000 and fsc_perp=1000 would take branch 2, branch 4, then branch 10, and be classified as nano.

Answer Questions 5, 6, 7

Step 5: Evaluate the decision tree on the test data.

Use the predict function to generate predictions on your test data. Then, compare these predictions with the class labels in the test data itself.

In R, if you write A==B and A and B are vectors, the result is a vector of 1s and 0s. The sum of this vector will be the number of correct predictions. Dividing this sum by the size of the test dataset will give you the accuracy.

Answer Question 8

Step 6: Build and evaluate a random forest.

Load the randomForest library, then call randomForest using the formula object and the data, as you did to build a single tree:

library(randomforest) model <- randomForest(fol, data=trainingdata)

Evaluate this model on the test data the same way you did for the tree.

Answer Question 9

Random forests can automatically generate an estimate of variable importance during training by permuting the values in a given variable and measuring the effect on classification. If scrambling the values has little effect on the model's ability to make predictions, then the variable must not be very important.

A random forest can obtain another estimate of variable importance based on the Gini impurity that we discussed in the lecture. The function importance(model) prints the mean decrease in gini importance for each variable. The higher the number, the more the gini impurity score decreases by branching on this variable, indicating that the variable is more important.

Call this function and answer Question 10

Step 7: Train a support vector machine model and compare results.

Use the e1071 library and call model <- svm(fol, data=trainingdata).

Answer Question 11

Step 8: Construct confusion matrices

Use the table function to generate a confusion matrix for each of your three methods. Generate predictions using the predict function, then call the table functions like this:

```
table(pred = predictions, true = testingdata$pop)
```

Answer Question 12

Step 8: Sanity check the data

As a data scientist, you should never trust the data, especially if you did not collect it yourself. There is no such thing as clean data. You should always be trying to prove your results wrong by finding problems with the data. Richard Feynman calls it "bending over backwards to show how you're maybe wrong." This is even more critical in data science, because almost by definition you are using someone else's data that was collected for some other purpose rather than the experiment you want to do. So of course it's going to have problems.

The measurements in this dataset are all supposed to be continuous (fsc_small, fsc_perp, fsc_big, pe, chl_small, chl_big), but one is not. Using plots or R code, figure out which field is corrupted.

Answer Question 13

There is more subtle issue with data as well. Plot time vs. chl_big, and you will notice a band of the data looks out of place. This band corresponds to data from a particular file for which the sensor may have been miscalibrated. Remove this data from the dataset by filtering out all data associated with file_id 208, then repeat the experiment for all three methods, making sure to split the dataset into training and test sets after filtering out the bad data.

Answer Question 14

Now, if you have read and understood these instructions, type "yes" in response to this question!

Mark as completed