University of Washington Bothell
Computing & Software Systems

Course title:	Introduction to Machine Learning
Course number:	CSS 490 / 590
Term:	Winter 2012
Instructor:	Jeff Howbert

Project 1

Date assigned:	Sat. Jan. 28, 2012
Date due:	Sat. Feb. 11, 2012, 10:00 PM

There are three parts to the project. All students (CSS 490 and CSS 590) will complete Parts 1 and 2. CSS 590 students will also complete Part 3.

This project is focused on feature selection, using:

- logistic regression as the classifier
- the UC Irvine heart disease dataset

Part 1. [10 points] Write a MATLAB function that uses a wrapper method for scoring feature subsets. The wrapper is run multiple times with different random partitions of the data into training and test sets, in order to achieve a less variable estimate of the score.

This part is really just a minor elaboration on the code in matlab_demo_05.m, which was demonstrated at the end of Lecture 5.

Requirements:

- Encapsulate your scoring code in a MATLAB function. A basic framework for a function, with input and output arguments, is given below. Name your function `featureSubsetScore`, i.e. use this name in place of *`functionName`* in the framework.
- Use the input arguments to the function to pass in:
    - A matrix holding a subset of features from the full heart disease data matrix. It is the responsibility of the calling routine to form this subsetted matrix, containing only the desired subset of features. All samples should still be present, however.
    - The full vector of class labels.
- Do the following 10 times, inside a `for` loop:
    - Reset the default random number generator with a new random seed (please use the sequence of seeds 1 through 10).
    - Randomly divide the data matrix into a training set with 200 samples and a test set with 70 samples, using randperm().
    - Train a logistic regression model on the training set.

    - Use the trained model to make class predictions on both the training and test sets, with a probability threshold of 0.5.
    - Calculate prediction accuracy for both training and test sets. These comprise the basic scores, or metrics of quality, for the current subset of features. Use these formulas:
        - `trainScore = nTrainCorrect / nTrain;`
        - `testScore = nTestCorrect / nTest;`
    - Save these scores to use in further calculations after the `for` loop is finished.
- Calculate the mean of the scores over the 10 passes of the `for` loop, for both training and test sets.
- Return these two means as output arguments of the function.

Function framework:

```
function [ outArg1, outArg2, ... ] = functionName( inArg1, inArg2, ... )

      < do things with the values of the input arguments >
            …
      outArg1 = < some result you want to return >
      outArg2 = < some other result you want to return >

end
```

NOTE: In MATLAB, input and output arguments of functions are not typed (e.g. as `int`, `double`, etc.) as they are in C++. Furthermore, there are no restrictions on the types of the input arguments – they can be numbers, logicals, vectors, matrices, strings, cell arrays, structures, etc. – except that what is actually passed in must be compatible with how you try to process it inside the function. There are likewise no restrictions on the types of the output arguments.

Save your function as a file named `featureSubsetScore.m` in your current MATLAB working directory. Then execute the following code from the command line:

```
clear all;
load heart;
format compact;
% NOTE: no semicolons at the end of the next two lines
[ trainScoreMean, testScoreMean ] = featureSubsetScore( dat( :, [ 1 : 13 ] ), label )
[ trainScoreMean, testScoreMean ] = featureSubsetScore( dat( :, [ 1, 3, 5 ] ), label )
```

Assuming you faithfully observed the requirements above, and took full advantage of the code in `matlab_demo_5.m`, your output should be:

```
trainScoreMean =
    0.8660
testScoreMean =
    0.8171
```

```
trainScoreMean =
    0.7280
testScoreMean =
    0.7143
```

(I'm telling you the correct answer so we can both be sure this part works before you move on to Parts 2 and 3.)

For Part 1, turn in the following:

- The file `featureSubsetScore.m` containing your function.
- The console output from the two tests of the function (cut-and-paste into a document).

Part 2. [30 points] Write a script (not a function) that generates 1000 random feature subsets of a specified size *k*, scores each subset with function `featureSubsetScore`, and outputs to the console:

- the best mean test score from the 1000 trials
- the feature subset corresponding to the best score

Make sure you reset the default random number generator with a new random seed before each random feature selection (please use the sequence of seeds 1 through 1000).

Use the pseudocode in `random_selection.pdf` as a guide to structuring your script. You shouldn't need more than about 20 lines of code, not counting output statements. Save your script as `randomSelection.m`.

Run your script with feature subset sizes from *k* = 1 to 7. It's fine to change this parameter by hand in the editor.

For Part 2, turn in the following:

- The file `randomSelection.m` containing your script.
- The console output (best score and feature subset) from each of the 7 runs (cut-and-paste into a document).
- Answers to the following questions:
    - Is there a value of *k* at which the best mean test score stops improving?
    - How does the score at that value of *k* compare with the best mean test score obtained by running `featureSubsetScore` on the full set of features?
    - What can you conclude about the information content of the full set of features?

Part 3. [40 points] Write a script (not a function) to do a greedy forward selection of features. Use the pseudocode in `forward_selection.pdf` as a guide to structuring your script. You shouldn't need more than about 30 lines of code, not counting output statements. Save your script as `forwardSelection.m`.

Run your script with the target number of selected features *k* set to 7. The script may terminate before it reaches this number of selected features (review the termination conditions in the pseudocode to make sure you understand why). The script should output to the console:

- the best mean test score at each intermediate value of *k* (*k* = 1, 2, 3, etc.)
- the feature subset corresponding to the best score at each intermediate value of *k*

For Part 3, turn in the following:

- The file `forwardSelection.m` containing your script.
- The console output from running the script (cut-and-paste into a document).
- Answers to the following questions:
    - Compare the value of *k* at which forward selection terminates to the value of *k* at which the best mean test score stops improving in Part 2. How are they similar or different?
    - Compare the best feature subset at the termination of forward selection to the best subset for the value of *k* at which the score stops improving in Part 2. How are they similar or different?
    - What do you conclude about the relative value of forward vs. random selection for finding a high-scoring subset of features from this dataset?