Homework Assignment #2
Due at midnight on Tuesday, 10/25

Part 1

For this part of the homework, you are to implement a *k*-nearest neighbor learner for both classification and regression.

Your programs should read files that are in the ARFF format. In this format, each instance is described on a single line. The feature values are separated by commas, and the last value on each line is the class label (for classification) or numeric response (for regression) of the instance. Each ARFF file starts with a header section describing the features, class labels, and numeric responses. Lines starting with '%' are comments. See the link above for a brief, but more detailed description of the ARFF format. Your program should handle numeric and nominal attributes, and simple ARFF files (i.e. don't worry about sparse ARFF files and instance weights). Example ARFF files are provided below.

Your programs should should implement a *k*-nearest neighbor learner according to the following guidelines:

- Assume that for classification tasks, the class attribute is named 'class' and it is the last attribute listed in the header section. Similarly, for regression tasks, the response is named 'response' and it is the last attribute listed in the header section.
- Assume that all features will be numeric.
- Use Euclidean distance to compute distances between instances.
- Implement basic *k*-NN. You shouldn't do distance weighting, edited nearest neighbor, k-d tree lookup, etc.
- Determine whether the given data set is a classification or regression task depending on whether the last attribute in the training-set ARFF file is named 'class' or 'response'.
- **If there is a tie among multiple instances to be in the *k*-nearest neighbors, the tie should be broken in favor of those instances that come first in the ARFF file.**
- **If there is a tie in the class predicted by the *k*-nearest neighbors, then among the classes that have the same number of votes, the tie should be broken in favor of the class comes first in the ARFF file.**

Your programs should be callable from the command line. The first program should be named `kNN` and should accept three command-line arguments as follows:
`kNN <train-set-file> <test-set-file> k`
This program should use the training set and the given value of *k* to make classifications/predictions for every instance in the test set.

The second program should be named `kNN-select` and should accept five command-line arguments as follows:
`kNN-select <train-set-file> <test-set-file>` k_1 k_2 k_3
This program should use leave-one-out cross validation with just the training data to select the value of *k* to use for the test set by evaluating k_1 k_2 k_3 and selecting the one that results in the minimal cross-validated error within the training set. To measure error for regression tasks, you should use mean absolute error.

If you are using a language that is not compiled to machine code (e.g. Java), then you should make scripts called `kNN` and `kNN-select` that accept the command-line arguments and invokes the appropriate source-code program and interpreter. See HW #1 for instructions on how to make such a script.

As output, your programs should print the value of *k* used for the test set on the first line, and then the predictions for the test-set instances. For each instance in the test set, your program should print one line of output with spaces separating the fields. For a classification task, each output line should list the predicted class label, and actual class label. This should be followed by a line listing the number of correctly classified test instances, and the total number of instances in the test set. For a regression task, each output line should list the predicted response, and the actual response value. This should be followed by a line listing the mean absolute error for the test instances, and the total number of instances in the test set.

You should test your code on the following two data sets:

- wine_train.arff, wine_test.arff
- yeast_train.arff, yeast_test.arff

Here are the outputs your first program (`kNN`) should produce (where *k* is specified as an argument):

- wine data set with *k*=3
- wine data set with *k*=5
- yeast data set with *k*=3 **(updated 10/18)**
- yeast data set with *k*=5 **(updated 10/18)**

Here are the outputs `kNN-select` program should produce when run with $k_1=3$, $k_2=5$, $k_3=7$:

- yeast data set
- wine data set

The first data set is for a task that involves predicting wine scores from chemical properties. The second data set is for a task that involves predicting the compartment in a cell that a yeast protein will localize to based on properties of its sequence.

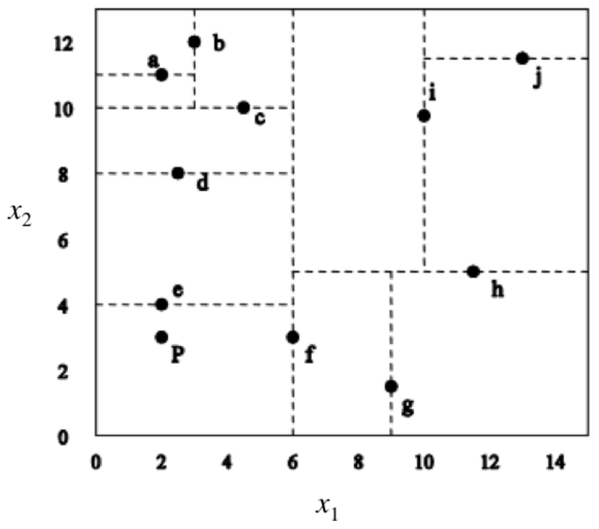
Part 2

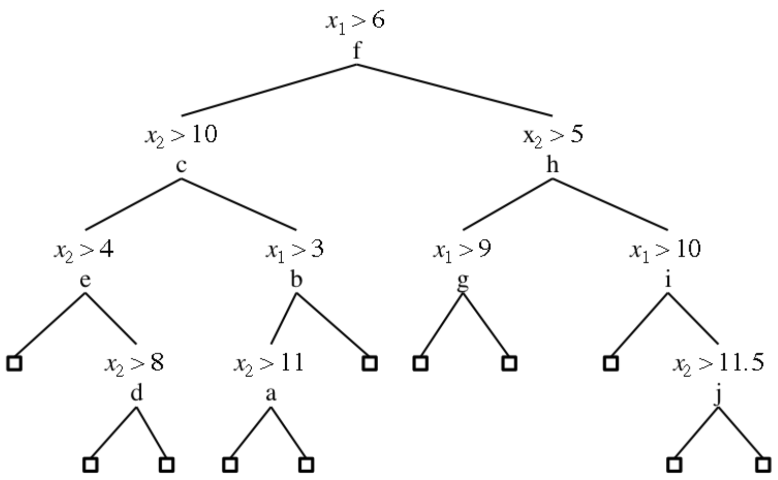
For this part you will explore the effect of the *k* parameter on predictive accuracy.

- For the yeast data set, draw a plot showing how test-set accuracy varies as a function of *k*. Your plot should show accuracy for *k* = 1, 5, 10, 20, 30.
- For the wine data, draw a similar plot showing test-set mean absolute error as a function of *k*, for *k* = 1, 2, 3, 5, 10.
- For the yeast data set, construct confusion matrices for the *k* = 1 and *k* = 30 test-set results. Show these confusion matrices and briefly discuss what the matrices tell you about the effect of *k* on the misclassifications.

Part 3

Using the k-d tree and the training set displayed in the figure below, show how the nearest neighbor for $\mathbf{x}^{(q)} = (7, 10)$ would be found. Assume that, for all instances in the figure, the features (i.e. coordinates) have integer values. For each step in the search, show the distance to the current node, the best distance found so far, the best node found so far, and the contents of the priority queue. You should use Euclidean distance and assume that the coordinates of the instances in the figure below are those shown in the table below.





Instance	***x***	***y***
a	2	11
b	3	12
c	5	10
d	2	8
e	2	4
f	6	3
g	9	2
h	12	5
i	10	10
j	13	11.5

Submitting Your Work

You should turn in your work electronically using the Canvas course management system. Turn in all source files and your runnable program as well as a file called hw2.pdf that shows your work for Parts 2 and 3. All files should be compressed as one zip file named `<Wisc username>_hw2.zip`. Upload this zip file as Homework #2 at the course Canvas site.