```
CS578 Fall 2002
Empirical Methods in Machine Learning and Data Mining
Homework Assignment #2
Due: Tuesday, October 29 2002

The goal of this assignment is to implement k-nearest neighbor (kNN)
and run a few experiments with kNN and cross validation.  You can use
any programming language you want, but faster languages such as C will
make it easier to run the experiments.

Here are some features the code should support:

 - should work for different values of k
 - should work for both classification and regression problems.
   regression is very easy -- just average (or weighted average)
   of the values; classification requires a little more work since
   you have to count how many of the nearest neighbors are in each
   class.
 - the distance function should support feature weights, but it
   does not have to optimize those feature weights -- see extra
   credit below.  for most of your experiments you should set
   the feature weights to 1.0
 - should support leave-one-out-cross-validation (LOOCV) and
   report the LOOCV accuracy and RMSE
 - should support kNN (prediction is the average or predominant
   class of the k-nearest neighbors), weighted kNN (prediction
   is based on the k-nearest neighbors, but weighted by their
   distance to the test case), and locally weighted averaging
   (prediction depends on all cases in the train set, not just
   the nearest k, weighted by their distance to the test case).
   note that these are so similar to each other you can write just
   one loop that does all three.
 - in classification problems, should calculate a probability that
   the test case is in each of the classes

EXPERIMENTS:

 1: We have put a dataset on the web for you to use.  It contains
    15,000 cases and 144 attributes.  From this data you will
    predict a boolean variable (just two classes).  The class to
    predict is the last column in the dataset.

 2: Using LOOCV, experiment with different values of k to find
    which k works best.  Do this using unweighted kNN (i.e., the
    distance is not taken into account).

 3: Vary the size of the training set to create learning curves
    for different values of k.  Does the best value of k change
    when the train set size changes?

 4: Instead of kNN, switch to locally weighted averaging where you
    us all cases in the train set, but their predictions are weighted
    inversely by distance.  Use weights that fall off exponentially
    with distance.  Experiment with different values for the kernel
    width which controls how quickly the vote of a training case falls
    off with distance.  Which is best?  Does the best kernel width
    change with the size of the training set?

 5: Weight features with weights equal to 1/variance or 1/(max-min).
```

This helps compensate for features that have unusually small or large ranges or which vary much more or much less than other features. Does performance improve when you do this?

EXTRA CREDIT -- do one or more of the following::

- experiment with kNN on the thyroid data from hw1 which has more than two classes.
- try to optimize the feature weights to improve performance by applying some form of numerical optimization to the weights or by using something like information gain to scale the weights
- do feature selection to find a subset of the features that seems to perform better than using all the features
- implement and test locally weighted regression using simple local models such as linear regression.
- experiment with different distance functions for locally weighted averaging or weighted kNN
- do experiments with weighted kNN to find good values for k and the kernel width
- make the code more efficient, document the methods you use and show that there is a significant the speed improvement
- implement and experiment with n-fold cross validation
- modify your code so it uses a final test set that is held out of the train set used for LOOCV. once you find the best value of k or kernel width using LOOCV on the train set, report the performance of this model on the final test set
- change the distance function to take into account different types of variables (e.g., boolean, nominal, ordinal, integer, and continuous). explain how you compute distance on each type, and how you combine the distances on different types into one total distance. does performance improve when you do this?

Hand in a brief summary of the results with enough documentation so that we can see what you did and how you did it. Do not write a paper -- this is homework, not a project. You might want to use your kNN code later in the class project, so effort spent now to become familiar with kNN and write good code should pay off later. Note that kNN and LOOCV with large training sets is computationally expensive: you want to choose a good programming language and write efficient code. Don’t wait until the last minute to run the experiments.

Have fun.