**a.** Not quite, unless we claim that k=1. This program divides the data into a training and test set, trains the model once using said training set, then predicts the classes using the test set. If we instead utilized k=2 cross-validation folding as in part one, then there would be a fit to xfold1, yfold1 and a predict for xfold2; then a fit to xfold2, yfold2 and a predict to xfold1. That is, the DBN algorithm would execute the train-test step at least twice. I suspect the omission of k>1 here is simply due to the amount of time the program would require to run.

**b.** The training data consists of 80% of the total sample.

**c.** There are 360 instances in the test set.

**d.** The training data contains 1437 instances.

**e.** Each instance in the test set contains 64 features (the length of a single array, which in turn is a flattened 8x8 px image). In total, there are 64*360 = 23040 features in total comprising the test set.

**f.** Again, each instance contains 64 features. The total number of training set features, then, is 64*1437 = 91968.

**g.** From the README file, there are 10 classes.

**h.** Each class is an integer, ranging from 0 to 9. More explicitly, the classes are 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9.