Using datasets to make predictions

Learning objectives

This exercise is designed to help you learn about (and assess whether you have learned about):

- Reading from CSV files into parallel arrays
- Finding the minimum value in an array
- Using the `Arrays.sort` function in Visual Basic
- Using nested loops

Background

These lab exercises use datasets containing information about three different varieties of iris flower: setosa, versicolor & virginica. One set of data (from which you've seen a sample in Lab 03), FisherIrisDatasetValues.csv & FisherIrisDataset-Labels.csv, created by Sir Ronald Aylmer Fisher from measurements of actual plants will be used as a basis for predicting the variety of new sample plants by comparing the measurements of the new sample with the verified data from the dataset. File FisherIrisDatasetValues.csv contains 120 flowers (i.e. rows) each containing four numeric measurements (i.e. columns): sepal length, sepal width, petal length and petal width. The flower type of each of these flowers is recorded in file FisherIris-DatasetLabels.csv.

The other set of data, OtherIrisSamplesValues.csv & OtherIrisSamplesLabels.csv, contain information on 30 more flowers and will be used as a test of the reliability of the prediction system that you will build in the first exercise.

Instructions

In order to compare the measurements of say, the petal length of the new sample, to those in the database, you need to calculate a similarity measure. A common way to do this is to calculate the absolute difference between the sample and the verified data for each of the measurements and compute a total difference measurement. Rather than just computing the raw total difference between the measurements, it is more accurate to compute each of the differences as a fraction of the range of values for each measurement. This distance measure is known as the Manhattan Distance (a.k.a. City-block distance). For example, if the length of the petal was two inches longer than one sample in the dataset, the contribution of two inches to the total difference might out-weigh a difference of one half inch in the petal width, whereas maybe the difference in the petal width is proportionally much more of a difference in the typical plant measurements. The entries from the dataset that have the smallest difference from the sample are then used to predict the iris type of the sample. This algorithm, called the kNN algorithm, or k-nearest neighbors, can be applied to many different types of problems from predicting your credit score to predicting the type of cancer in a patient. Use the algorithm below to develop a program, (MATLAB script), called Lab9Part1_KNN.m that will get four measurements of a single iris plant from the user, and from those measurements, predict the iris variety of that plant.

1. Read the data in the file `FisherIrisDataset.csv` into five parallel arrays called `sepalLength`, `sepalWidth`, `petalLength`, `petalWidth` and `labels`, respectively. There are a total of 120 iris flowers in the file, so your arrays should be sized appropriately.

2. After reading the data into parallel arrays, output the array values in a table formatted as follows:

```
Sample #   Sepal length   Sepal width   Petal length   Petal width        Label
--------   ------------   -----------   ------------   -----------   ----------
     XXX            X.X           X.X            X.X           X.X   XXXXXXXXXX
     XXX            X.X           X.X            X.X           X.X   XXXXXXXXXX
       .              .             .              .             .            .
       .              .             .              .             .            .
       .              .             .              .             .            .
-------------------------------------------------------------------------------
```

Declare a format string for such a table as follows:

```
Dim fmtStr As String = "{0,8:000}   {1,12:0.0}   {2,11:0.0}   {3,12:0.0}   " & _
    "{4,11:0.0}   {5,10}" & vbNewLine
```

3. Use the `InputBox` Visual Basic function to ask the user to input sepal length, sepal width, petal length and petal width values for their new iris sample and assign those values to four variables called `userSepalLength`, `userSepalWidth`, `userPetalLength` and `userPetalWidth`, respectively.

D level – Echo the values from the file to `outResults` in a nicely formatted table

---

1. Declare another array, called `distances`, that will hold the distance computed between the iris flower sample input by the user, and each of the iris flower samples read from the file `FisherIrisDataset.csv`.
2. After computing the distances between the user's iris flower sample and those in the dataset, output the distances in a table formatted as follows:

```
Sample #   Distance
--------   --------
     XXX        X.X
     XXX        X.X
       .          .
       .          .
       .          .
-------------------
```

NOTE You should be able to use the value of the `fmtStr` variable from above to create an appropriate format string for this problem.

3. Using a `For`-loop, find the minimum value in the `distance` array. Be sure to keep track of the index where the minimum value appeared. Then, using the minimum distance and its corresponding index, make a prediction as to the class of the user's iris flower sample. For now, the prediction will be the label of the iris flower sample from the file that is the least different (i.e., minimum distance) from the user's sample. Output the predicted class in a message based on the following example:

```
The predicted class for the user's sample is 'versicolor'.
This prediction is based on dataset sample #XX which has a computed distance of X.X.
```

C level – Compute and output distances to `outResults` in a nicely formatted. Then make a prediction based on the minimum distance value that was computed.

As you may be able to guess, the simple model of find the dataset sample which is the least different from the user's sample and use its class to predict the class of the user's sample is not very robust, i.e., it is likely to produce incorrect predictions. However, your work up to this point is not for naught. Rather than using just the dataset sample that is least different from the user's sample, we can greatly improve the accuracy of our predictions by looking at the *k* least different dataset samples. However, as it turns out, finding the *k* least different dataset samples is more complicated than find the least different dataset sample, so as before, we will work through it in steps. The basic idea is to sort the `distance` array and then select the first *k* values from the array as the will represent the *k* minimum distances.

1. To see how this works, we must first sort the `distance` array. To do this, we will rely on a built-in function in Visual Basic, conveniently named `Array.Sort`. So, to sort the `distance` array, you would say:

```
Array.Sort(distance)
```

   Output the sorted `distance` array to `outResults` to see that it is in fact sorted.

2. Unfortunately, using the above line of code, after sorting the `distance` array we have no way to determine which dataset sample each of the distance values belonged to. Fortunately, with a small modification, we can have `Array.Sort` produce a second array which stores the sample number for each value in the sorted `distance` array. To do this, we must supply a second array to the `Array.Sort` function. We will call this array `sampleNumber` and it will contain the sample number that is associated with each distance value.

```
Array.Sort(distance, sampleNumber)
```

   Now we can find out which dataset sample belongs to each distance in the `distance` array like so:

```
outResults.AppendText("The distance between sample #" & sampleNumber(0) & _
    " and the user's sample is " & distance(0))
```

3. Reproduce the prediction from the previous part, i.e., predict based on the class of the dataset sample that is the least different from the user's sample. This time however, use the sorted array instead of computing the minimum distance in a loop. Output your prediction using the same statements as above.

4. Ask the user to input an integer, *k*, that will be used to choose the *k* closest matches to the user's sample. Using the sorted `dataset` array and the `sampleNumber` array, output the *k* least different dataset samples in a table formatted as follows:

B level – Added `sampleNumber` array, sort parallel arrays, and make prediction using the minimum distance sample. Then output a table with the *k* minimum distance samples.

---

1. Calculate the majority type in the top k and display your prediction. PS: you need to find out the count of each type of flower which exits in the top k. The type associated with the largest count would be your final prediction.

Based on Lab09 from CSCI 140.

- The problem will be the following:
  - You are given a file with a user ratings for X number of movies, no missing entires.
  - Your job is, given the ratings for X-1 movies for a user, what do you predict that they will rate the movie with no rating
  - To do this, you will find the user who is most similar to the user for the movies that do have ratings, then use their rating for the missing movie as the rating for the query user.
  - https://grouplens.org/datasets/movielens/
    - Use one of these datasets, choose columns and users to create a matrix with no empty entries.
  - https://www.kaggle.com/c/predict-movie-ratings