Using datasets to make predictions

TL;DR

Implement a program to predict the rating that a user will assign to a movie based on their ratings for other movies and the ratings of other users for the same movies.

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
- More practice with formatting strings
- Finding the minimum value in an array
- Using the `Arrays.sort` function in Visual Basic

Background

These lab exercises use a dataset that is maintained by the GroupLens research group at the University of Minnesota. The dataset is part of the MovieLens project. In their own words, MovieLens is a platform for "*Non-commercial, personalized movie recommendations*"; think Netflix recommendations, but not for profit. The dataset that we will be exploring consists of movie ratings for five popular movies from 22419 unique users. In the file `ML-latest.csv` each row constitutes the ratings for a single user for each of the five movies. For example, the first three rows in the file look like:

```
4.0,4.5,4.0,4.0,3.0
3.0,4.0,5.0,4.0,5.0
5.0,5.0,5.0,5.0,5.0
```

This means that user 0, rated movie 1, 4.0 stars out of 5 possible, movie 2, 4.5 stars, movie 3, 4.0 stars, and so on. Likewise for user 1 and 2. The title for each of the five movies is recorded in file `ML-titles.txt`.

Instructions

The objective for the lab is to make an accurate prediction of the rating that a user will assign to a movie based on their ratings for other movies and the ratings of other users for the same movies. The basic idea is to take a user's ratings for a set of movies and find other users with similar ratings. These similar users can then be used to inform our prediction.

In order to identify similar users, we must have some quantitative way to determine similarity, i.e., a *similarity measure*. A common way to do this is to calculate the absolute difference for each of the movie ratings for two users and add these together to compute a total difference measurement. This distance measure is known as the Manhattan Distance (a.k.a. Taxicab distance).

Take the first two users in the example above. The difference between them is calculated as:

```
  |4.0-3.0| + |4.5-4.0| + |4.0-5.0| + |4.0-4.0| + |3.0-5.0|
= 1.0 + 0.5 + 1.0 + 0.0 + 2.0
```

= 4.5

So we would say that the Manhattan distance between the two users is 4.5. It is important to note that in this case, *distance* is the opposite of *similarity*, which is what we are really after. So, to find the most similar users, we should find the users with that have the least distance between them.

1. Give it a try. Fill in the following table by computing the distance between the three users in the example above, then determine which two users are the most similar. Note, it is not necessary, or meaningful in this case, to compute the distance between a user and themselves, so each of these entries in the table is marked with `---`. You should also note that the Manhattan distance is *symmetric*, i.e., the distance between user X and user Y is the same as the distance between user Y and user X, so you only need to compute this distance once and then can fill in both cells in the table.

User	0	1	2
0	---		
1		---	
2			---

2. **TODO** Read a small file with a single column

```
outResults.Clear()
outResults.AppendText(totalUsers & " users' ratings were read from the file.")
```

3. **TODO** Read a small file with all columns
4. Read the data in the file `ML-ratings.csv` into five parallel arrays called `movie1`, `movie2`, `movie3`, `movie4` and `movie5`, respectively. There are a total of 22149 users (rows) in the file, so your arrays should be sized appropriately.
5. After reading the data into parallel arrays, output the array values in a table formatted as follows:

```
User ID   Movie 1   Movie 2   Movie 3   Movie 4   Movie 5
-------   -------   -------   -------   -------   -------
    XXX       X.X       X.X       X.X       X.X       X.X
    XXX       X.X       X.X       X.X       X.X       X.X
      .         .         .         .         .         .
      .         .         .         .         .         .
      .         .         .         .         .         .
-----------------------------------------------------------
```

   Declare a format string for such a table as follows:

```
Dim fmtStr As String = "{0,7}   {1,7:0.0}   {2,7:0.0}   {3,7:0.0}   " & _
    "{4,7:0.0}   {5,7:0.0}" & vbNewLine
```

   which could then be used to format six variables like so:

```
String.Format(fmtStr, i, movie1(i), movie2(i), movie3(i), movie4(i), movie5(i))
```

   Thus, in `fmtStr`, we say that there are six *format items*, one for each of the six columns in the table. Each format item is demarked by `{...}`. Within the curly braces, the first number indicates the *value index*, i.e., which value in the line `String.Format(...)` is to formatted. The first value after the format string, `fmtStr` in this case, is

considered value 0. So, `{0,7}` is the format item for the value `i`, `{1,7:0.0}` for value `movie1(i)`, and so on. After the value index is the *alignment*, i.e., how wide the formatted column should be, as well as whether it should be left or right justified. So, `{0,7}` indicates that value `i` should be formatted as seven characters wide and right justified. You can see that in this format string, all format items are seven characters wide and right justified. Lastly, following the alignment is the *format specifier*, which indicates out a value should look when formatted. There are many possibilities for a format specifier, so we will not go into the details. Suffice it to say that the `0.0` format specifier in `{1,7:0.0}` means that value `movie1(i)` should be formatted as a decimal value with a single digit following the decimal. If you would like more information on any of these topics, see this documentation from Microsoft.

6. **TODO** Compute and display average rating for each movie.

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

---

1. Use the `InputBox` Visual Basic function to ask the user to ratings for movie 1, movie 2, movie 3 and movie 4. Assign these values to variables called `queryMovie1`, `queryMovie2`, `queryMovie3` and `queryMovie4`, respectively. These values will represent the ratings for the hypothetical user for whom we are trying to predict a rating for movie 5.
2. Declare another array, called `distances`, that will hold the distance computed between the hypothetical user whose ratings for movies 1 through 4 were input in the previous step, and the users whose ratings were read from the file `ML-ratings.csv`.
3. **TODO** Compute distance between user 0 from file and hypothetical user.
4. **TODO** Compute distance between all users and hypothetical user, storing results in `distance` array.
5. After computing the distances between the hypothetical user and those users from the file, output the distances in a table formatted as follows:

```
User ID   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.

C level – Compute and output distances to `outResults` in a nicely formatted.

---

1. **TODO** Better description with reference to book on how to find minimum. 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 most similar user was user #XXX and the distance calulcated was X.X.
   The predicated rating for movie 5 is X.X.
   ```

2. 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.

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.

3. **TODO** Have students work on student name problem to see how the two parameter variant of `Array.Sort` can help them.
4. 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))
```

5. 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.
6. 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:

Then make a prediction based on the minimum distance value that was computed. 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