CS3240: Homework 1

Python 3.4 or higher should be used for this assignment

Deadline and submission rules: See Collab for the due date and time.
Collaboration rules:
You must **write the code and answer the report questions on your own**. You may only talk to other students about questions related to understanding the problem, i.e. what we want you to do. But you may not talk about how to solve it, either about design or about coding. You may not talk or get help from anyone outside the class other that the TAs and instructor, including students who are not in the class this term.

The following is a simplified version of a problem in statistical pattern recognition or data mining. The problem is called the k-nearest neighbor (k-NN) problem. This is a method for doing classification, which is a problem where each of a large set of data items has a vector of data values assigned to it, and each data item belongs to a category. A data item of unknown classification must be "classified", i.e. assigned to one category based on the known classification of the previously-classified data items. If k = 1, this is called the nearest-neighbor method.

An example: medical data values (e.g. weight, BMI, family history, etc.) are recorded for a large number of people when they were 40 years old. The two categories are whether or not they developed diabetes by the time they turned 65. For new 40-year-old patients, we want to predict if they will develop diabetes or not. We use the medical data values for each new patient and classify it based on the large data set of known classification.

Problem specification:
A large number of M data items are represented by a pair of values (X,Y) and a label. The label represents one of a fixed set of categories. (For this problem, we'll say there are just two categories, and your solution can assume that. In this explanation, we'll call these *cat1* and *cat2*, but the data file could contain different strings for the category names.) A new data item with its own (X,Y) value is to be processed against each of these data items and the Euclidean distance between the pair of (X,Y) values is calculated. We'll find the k items that are closest to the new data item and use those k items and their given category as described below. (Euclidean distance between two points is pretty simple, but if you've forgotten see Wikipedia.)

Program inputs:

- Value of k: prompt the user for a value of k.
- Value of M: prompt the user for the value of M, the number of values to be read from the data file. This could be smaller than the number of values actually found in the data file.
- Data file name: prompt the user for the name of a data file containing the classified data items. Each item will be on a line by itself, where each line is the category value (a string) followed by the X and Y values (all separated by 1 or more spaces). X and Y may be any floating point values (negative, positive or zero).
- Unclassified data values: prompt the user for (X,Y) value pairs. Keep prompting and processing these (see below) until the user enters values that are 1.0 and 1.0 (yes, this is kind of dumb but it's simple). **Clarification:** the user just enters two numbers, without any parentheses or commas.
  Example: `-3.2 1.7`

Calculations and outputs:
For each unclassified data-item that is entered, do the following:

1. Print the set of k data items that are closest to each unclassified data-item in non-decreasing order (i.e. nearest first). For each item, print its category, (X,Y) values, and distance to the unclassified data item on a separate line. (See example below.)
2. Print which category this data-item would be assigned to, based on whether the majority of the k-nearest-neighbors belong to the first or second category. This "voting" is how the k-nearest-neighbor algorithm classifies the data-item. This line can be in any format, as long as the last token on the line is the category-value.

3. Print the average distance of the k-nearest-neighbors to the data-item for each of the two categories. (This is not how k-NN makes a decision about classification, but your program should do it anyway.) Print one line for each category, and they can be in any format as long as the distance value is the last token on the line.

   Be careful about following these output specifications. We will attempt to grade your program's output automatically.

Example results:
Say that k=5, the unclassified data-item has value (0,0), and the nearest neighbors turn out to be:

```
cat2   2  0
cat1 0.5  0
cat2   1  0
cat1   0  0
cat2   0  3
```

Then the output for item (1) described above might look something like this:

```
cat1 0 0 0.0
cat1 0.5 0 0.5
cat2 1 0 1.0
cat2 2 0 2.0
cat2 0 3 3.0
```

The output for item (2) described above might look something like this:

```
Data item (0,0) assigned to: cat2
```

(This is because the "vote" for cat2 is 3-to-2 among the k=5 nearest neighbors.)

The output for (3) might look something like this:

```
Average distance to cat1 items:  0.25
Average distance to cat2 items:  2.0
```

Constraints:

- Use an object-oriented design for your solution. Use classes as appropriate. **Python 3.4 or higher should be used for this assignment.** As discussed in class and lab install Anaconda and PyCharm.
- Make use of standard libraries or other reusable components if this is useful.
- **Note!** Assume that this program could be used to process large data sets. Your design choices should take this into account.
- Document any error checking you do in the report (see below).
- Document any assumptions or changes you make in the report (see below).
- Comments are not required other than a header in the main file listing your name and contact info.
- Do not Zip any of your files. Just submit all of them to Collab as individual files. Note: If you find that the above specification unsatisfying for any reason, be aware of item 3 of the report you have to write (see below).

Report:
Submit a PDF or text file that has answers to the following questions ("the report"):

1. Name any *abstract data types* that are important in your program (if any).
2. Briefly describe any major *data structures* that you use and for what purpose in your program. If you used things from a standard library (or other reusable components), mention those here.
3. If you can think of any ways this problem statement is a bad requirements specification, briefly list at most three problems. This may be missing or confusing requirements. If you list anything here and you had to make an assumption or a change to make a working program, explain that.
4. Briefly describe your design in terms of the program components you are using. (We are deliberately not telling you exactly how to describe this. Describe your design in a brief but useful way -- imagine that you wanted a fellow student to understand how to do this without giving them your code.)

5. Briefly describe any error checking you did in your program. **Or could have done.**
6. Briefly describe three good test cases that you did (or should have done) to test that your code is correct. Say why your set of test cases are good choices.
7. (The answer to this part will not affect your grade at all. We're curious.) How many hours did you spend on coding this assignment? (Please be honest. We won't hold this against you in any way!)

How we'll grade this report:
We'll read your report pretty rapidly, looking to see if you have a good grasp of the high-level ideas we're looking for. We won't be reading for too much detail. We'll grade each part with a fairly simple grading-rubric:

5	4	3	2	1
knows it well	pretty good but could be better	acceptable but needs work	just below acceptable	well below acceptable

We really think you do not need long answers for these questions. If you find yourself writing a lot, try to make explain it more briefly. Also, it's important for this class that you communicate clearly and concisely, and that padding reports or documents with extra words makes for bad written communication.

A Picture Might Be Worth a Thousand Words:
In the picture below (from Wikipedia):

- The green circle is an unclassified point (one where the user has entered the X,Y values.
- The blue squares are points of one of the two categories, and the red triangles are points from the 2nd category. (These are read from the data file.)

Let's say k has value 3. Then of all the blue squares and red triangles, the ones inside the solid circle are the three nearest neighbors. In this case, 2 of the 3 are red triangles, so we'd classify the green circle as being in that category.

But let's say k has value 5. The five nearest neighbors are the ones insides the dashed circle. In this case, 3 of the 5 are blue squares, so we'd classify the green circle as being in that category.

Note: the circles are used to just show which ones are the k-nearest neighbors here. You are not calculating circles for this problem.