KNN implementation

YOUR TASK

Part of your homework for this week is to write out **pseudocode** for implementing KNN methods. The section "Pseudocode 101" below describes what pseudocode is. *Before we get there*, you will discuss the methods wtihin your group.

Part 1:

Discuss in your group how would you would implement predictions for **KNN classification**. Be as clear, thoughtful, and detailed as possible. This involves describing the algorithm step-by-step.

You may assume that all of our predictors are quantitative.

Some helpful discussion questions:

- What would be our input(s)? What do we want to output?
- What are the main components/parts/action for implementing this method? In other words, what do we generally need to "do"?
  - For each of the main components, what are the steps to achieve it? For example, if my hypothetical main action is "cook pasta", then the important steps might be to boil water, add salt *after* water is boiling, add pasta, set timer, then drain pasta.
- What information do we need to keep track of/store when implementing this method?

Part 2:

Next, discuss how you would implement **KNN regression**. *Recall that KNN regression is like KNN classification, but we have a quantitative response. So rather than predicting the majority class from the neighbor set, we predict the mean/average of the responses from the neighbor set.*

Part 3:

If you have time, you may begin writing pseudocode for either of these methods.

PSEUDOCODE 101

Pseudocode is more of an art than a science. It is a kind of structured english for describing algorithms. It allows the designer to focus on the logic of the algorithm without being distracted by details of language syntax (ideally, the pseudocode would be agnostic to the programming language, but sometimes it's easier to write it specific to your domain).

Pseudocode describes the entire logic of the algorithm so that implementation becomes a rote mechanical task of translating line by line into source code. I should be able to read your pseudocode and write working code from it. We use it here to demonstrate complete understanding of the method. In my mind, you don't truly understand how a statistical learning method works until you code it yourself!

When writing pseudocode, each specific action/piece of logic must be decomposed to the level of a single loop or decision. Very common constructs that we will use are for loops and if/else statements. Foor loops are specialized constructs for iterating a specific number of times. If/else statements evaluate binary outcomes. **It is always important to have indents such that reader can clearly see the conditions under which a line of a logic falls.** In the following examples, notice how and where I indent.

Examples

Suppose I have a vector of numbers `vec` and I want to obtain their sum. I could write the following pseudocode:

```
Set counter = 0, n = length(vec);

For i from 1, 2, ..., n:

  Add i-th value of vec to counter;

Return counter;
```

If maybe instead I wanted to obtain two separate sums, one of the even values and one of the add, I might write:

```
Set even_counter = 0, odd_counter = 0; n = length(vec);

For i from 1, 2, ..., n:

  If the i-th value of vec is even:

    Add the  i-th value of vec to even_counter;

  Else:

    Add the i-th value of vec to odd_counter;

Return even_counter and odd_counter;
```

Personally, I am fine with a little bit of actual code within the pseudocode, such as:

```
Set even_counter = 0, odd_counter = 0; n = length(vec);
```

```
For i from 1:n:

  If vec[i] is even:

    Add vec[i] to even_counter;

  Else:

    Add vec[i] to odd_counter;

Return even_counter and odd_counter;
```