Code Review - Week 3: Team Sirius kNN.py

Reviewer: Spotlight (Jaime Sanchez, Karan Singhal, Stefan Dimitrov)

Reviewee: Team Sirius (Brett Fischler, Hunter Wapman, Daniel Griffin, Tyler Lubeck)

Overview

We review the kNN algorithm used by team Sirius for their indoors location service. This is the meat of the project as it implements the actual means for determining the user's location by means of Wi-Fi access points' MAC addresses. Overall the code is well documented, very modular, easy to read and concise.

You could benefit from implementing unit tests for the individual functions that compose the algorithm. This will ensure your algorithm yields accurate and correct results, and will help with ensuring tweaks don't break things down the road. Moreover, having a way to measure the accuracy of the algorithm and compare it to future iterations will help improve your learning algorithm.

General

Does the code work? Does it perform its intended function, the logic is correct etc.

We weren't able to test the code since we do not have access to the access point data.

Is all the code easily understood?

The code in kNN.py is readable and modular, however, it could be useful to have the formulas being implemented written out in a comment at the beginning of the file.

Does it conform to your agreed coding conventions? These will usually cover location of braces, variable and function names, line length, indentations, formatting, and comments.

The code does conform to the python conventions. You even use docstrings to annotate functions which is great.

Is there any redundant or duplicate code?

There is some duplicate code in def euclideanOld(aps1, aps2). Seems like it's just euclidean without penalty applied. Should probably be removed.

Is the code as modular as possible?

The code is fairly modular which is great.

Can any global variables be replaced?

The only global variables are constants used for calculations with the exception of MAC_COUNTS. It looks like it might go and there's a TODO mentioning this, so kudos for that.

Is there any commented out code?

Yes there is. Looks like code to test the algorithm. Should go into a separate test.

Do loops have a set length and correct termination conditions?

Yes, however, a functional approach using map might be cleaner and safer.

Can any of the code be replaced with library functions?

In euclidian you calculate the union and set differences for aps1 and aps2. Are you sure there is no Set structure in python with implements those respective operations.

Can any logging or debugging code be removed?

None I could find.

Security

You do not check the testdata for validity but it comes from your db so perhaps you have those checks there instead.

Documentation

Do comments exist and describe the intent of the code?

Yes, they describe things very well, however a top level comment outlining the workings of the algorithm would be good.

Are all functions commented?

Is any unusual behavior or edge-case handling described?

Is the use and function of third-party libraries documented?

N/A

Are data structures and units of measurement explained?

It would be good to briefly describe the schema of the data you perform the calculations on.

Is there any incomplete code? If so, should it be removed or flagged with a suitable marker like 'TODO'?

Good use of TODO for MAC_COUNTS.

Testing

There is testing code at the end of the algo. Move this to a separate file. How do you intend to test the accuracy of the clasifier? It would be good to create a testing framework so you can compare further iterations and tweaks.

Do unit tests actually test that the code is performing the intended functionality?

No unit tests for individual functions would be good to cover that so you ensure the algorithm is implemented correctly.

Are arrays checked for 'out-of-bound' errors?

Nope, but that would not be necessary as they are consistent with their loop bounds and use the builtin range.

Could any test code be replaced with the use of an existing API?

This would be tricky as this is a learning algorithm and any testing would be done on the accuracy of the data.