Test Driven Development is essential for good data science. Here's why.

Kari Dempsey
Oct 19, 2017 · 4 min read

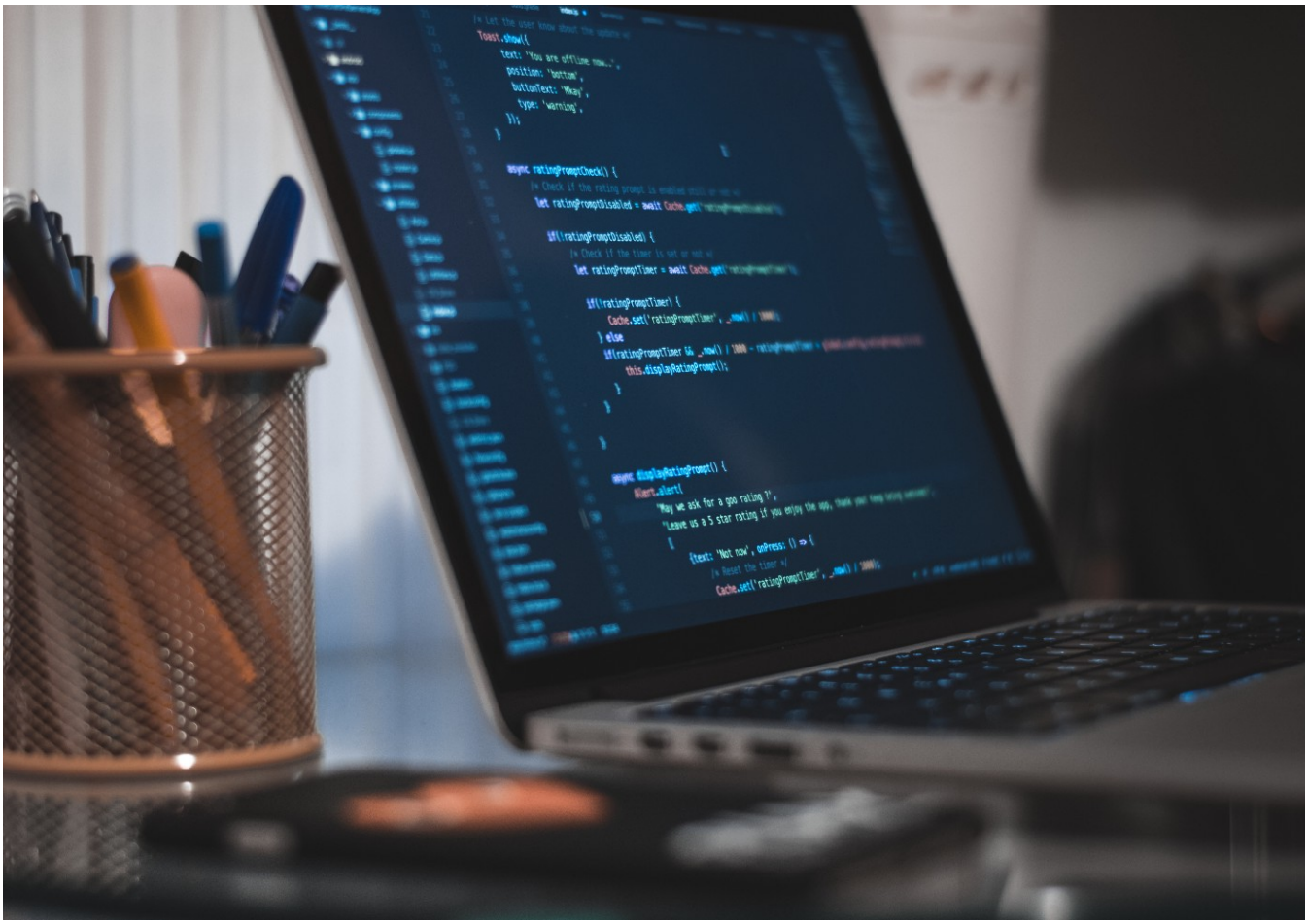

Photo by Fabian Grohs on Unsplash

There are many fun things about data science/analysis:

- figuring out how to solve tricky problems
- writing code
- communicating your awesome results

Things which fall into the less fun category:

- validating results on large data sets and complicated analysis without being able to easily prove your code actually does what you think it does
- piecing together old disparate pieces of code when asked to repeat analyses, that you thought was a one off

tl;dr Although test-driven development (TDD) doesn't make these problems magically go away, we believe it helps… considerably.

Our road to TDD

I head up the growing Data Science team at the UK Hydrographic Office. As a Data Science team, we have tried to take the best bits from software engineering. Over time we have adopted; agile, coding standards, pair programming, version control and testing. Latest to the party has been test-driven development (TDD), but it is probably the one thing that has made the biggest difference to how we work.

We work with many different teams/users on varied data sets; it has been common for us to have the following scenario's:

User: "We have this great data set we think you must be able to find out <insert interesting but badly defined problem>"

Us: "OK we'll take a look"… after some analysis we return to users with questions, "we found this, is it useful?"

User: "Looks interesting but what we really want to know is <insert well defined problem with clear value to business> "

Or

User: "Oh you remember that analysis you did for me a while ago, can you just re-run that for today's data, that should be quick right?"

Us: "Ah mmm, I thought you said that was a one off, OK no problem"…, Painstakingly piece the code back together and run. The data sets can be large and this type of spot checking after coding is not fun.

How does TDD help?

Test-driven development, TDD, is an approach which puts testing at the heart of your work. In its simplest form you write a test for your code first, make sure it fails then go about creating just enough code to pass the test, introduce more tests which fail, and repeat.

We found that formulating our thoughts by writing meaningful tests before setting about solving the problem helped clarify the boundaries of the problem, and how we could go about solving it. Retrospectively, before TDD it is surprising how little we really had thought about problems before we embarked on solving them.

We could have just added tests to help with some of the problems above but we quickly realised that retro-fitting testing to code is much harder than just writing the code with the tests there first. The process of red, green, refactor helped make our code more modular which means we can reuse more of it for different projects. We have also become much faster at implementing quality code.

Writing tests influences how you write your code, making it more legible to others, and your future self. It also allows you to refactor code with confidence you've not broken anything else.

Can I use TDD all the time?

No, but nearly. Exploratory data analysis is an example of when TDD doesn't really make sense. Once past this point the questions can be defined, perhaps more often than you think, and writing the tests can help you to do this more effectively. Having a defined idea of what you are expecting and how you might structure the code is a really useful process.

Isn't this obvious? (..says the software engineer)

Apparently not. Although it is becoming more common to hear about data science teams doing this, from those we meet the majority do not use these techniques or even employ code versioning, coding standards, tests, etc. For most, our experience is that data scientists know about code versioning, standards, tests and think these are things they should be doing, but are quite often not doing. But TDD, perhaps, is something that is not even on their radar.

We think TDD has helped us:

- improve our code quality and reusability
- improve the speed which we get to the root of the problems which have real value
- make data science less painful more fun, as we can focus on the problem solving rather than debugging

One area we are still developing is our TDD and general testing for machine learning algorithms in production, perhaps the topic for a future blog post… any thoughts welcome.

Contact kari.dempsey@ukho.gov.uk