Convexity in Predictive Value & Why Your Tests Are Flaky.

20 May 2020 · agile, automation, python

A long time ago, in a country far away, a cunning politician suggested a way to reduce crime. He stated that a simple test that could be used to catch all the criminals. When tested, all the criminals would fail the test and be locked up. There'd be no need for expensive courts, crooked lawyers or long drawn out trials.

The politician failed to give details of the test when pressed by journalists, stating that the test was very sensitive and they wouldn't understand it. His supporters soon had their way and the politician was elected to office.

On his first day in office, he deployed his national program of criminality-testing. Inevitably the details of the test leaked out. The test was simple and was indeed capable of ensuring 100% of criminals were detected.

The test was: If the person is alive, find them guilty and lock them up.

The test had a sensitivity of 100%, every single actual… real… bonafide criminal would fail the test and find themselves in prison.

Unfortunately, the test was not specific. Its specificity, found after an extensive and thorough review, was 0%. All the people who were definitely not criminals also found themselves 'guilty' and were sent to prison.

> …only checking if a feature is present and 'working'. This soon results in a preponderance of tests that fail intermittently.

In medicine, they use "sensitivity" and "specificity" to describe the accuracy of medical tests. Combined with the details of the disease prevalence (the proportion of people that actually have the disease or in our case the % of criminals in the population) clinicians can calculate the Predictive Values of a disease.
Positive & Negative Predictive Value are ways to summarise the usefulness of a test. A high Positive Predictive Value would mean that the majority of people who tested positive for a disease actually had the disease and weren't victims of a false alarm (https://en.wikipedia.org/wiki/False_positives_and_false_negatives#False_positive_error) from a dodgy diagnostic test.

In software development a flaky test, that is one with low Positive Predictive Value (PPV) can be a useful entry point into how the app or tests are functioning. It's the sort of messy real-world situation that can illuminate the emergent behaviour of a complicated system.

> ...naive practitioner often attempts to increase a team's test-automation levels by encouraging scenario testing, focusing on checking if a feature is present and 'working'

But for creating reliable tests, suitable for a continuous integration system, we need tests with a high positive predictive value. As our electorate (above) found, not finding out the specificity of the test can come back to haunt us. Not having a high specificity can land us with a lot of false alarms.

For a fairly rare bug that might cause a failure 5% of the time, you need to be sure not to lower the test specificity. The reason for this is a convex relationship between specificity and Positive Predictive value when we maintain high sensitivity.

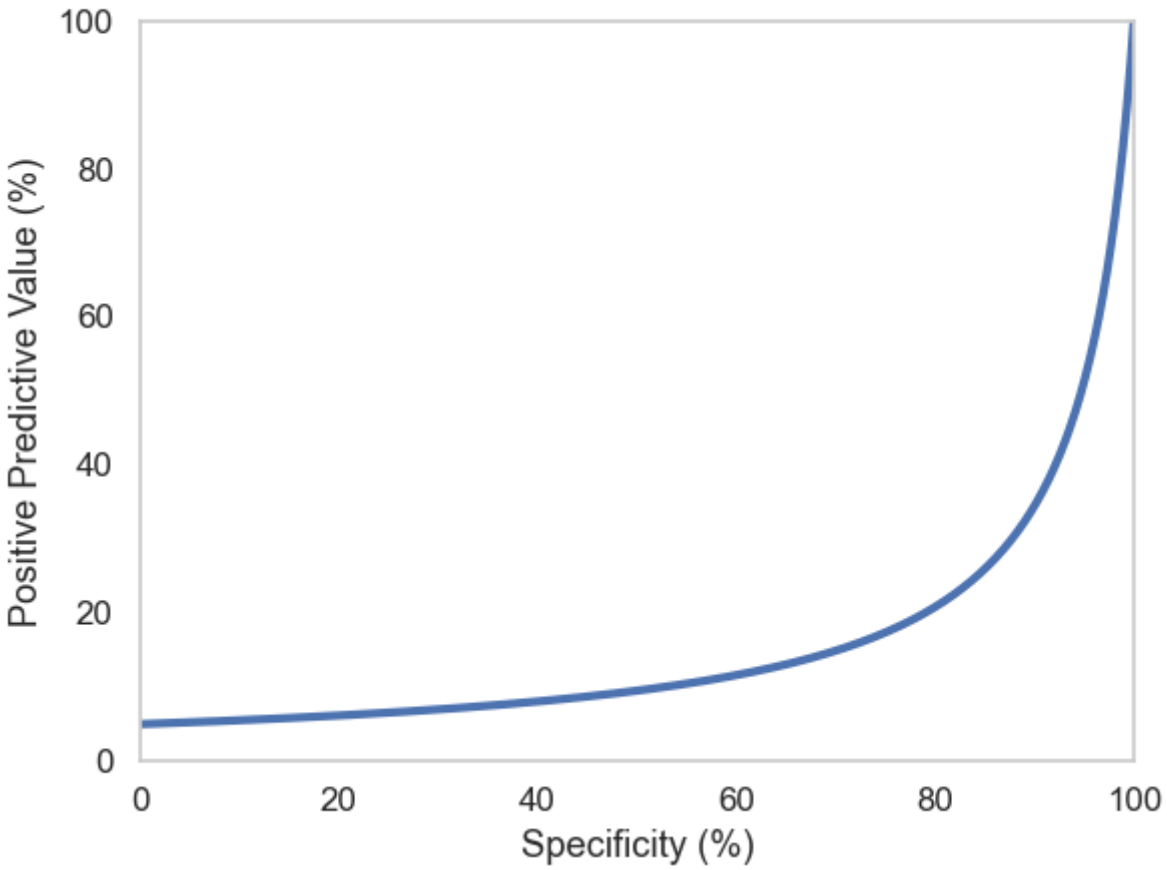


Figure 1: The convex relationship between Specificity and Positive Predictive Value is important when choosing to focus your team's time. Not ensuring your tests are highly specific will tend to cause a disproportionate number of unhelpful failures.

While conversely, the relationship between sensitivity and Positive Predictive Value is concave, given a high specificity.

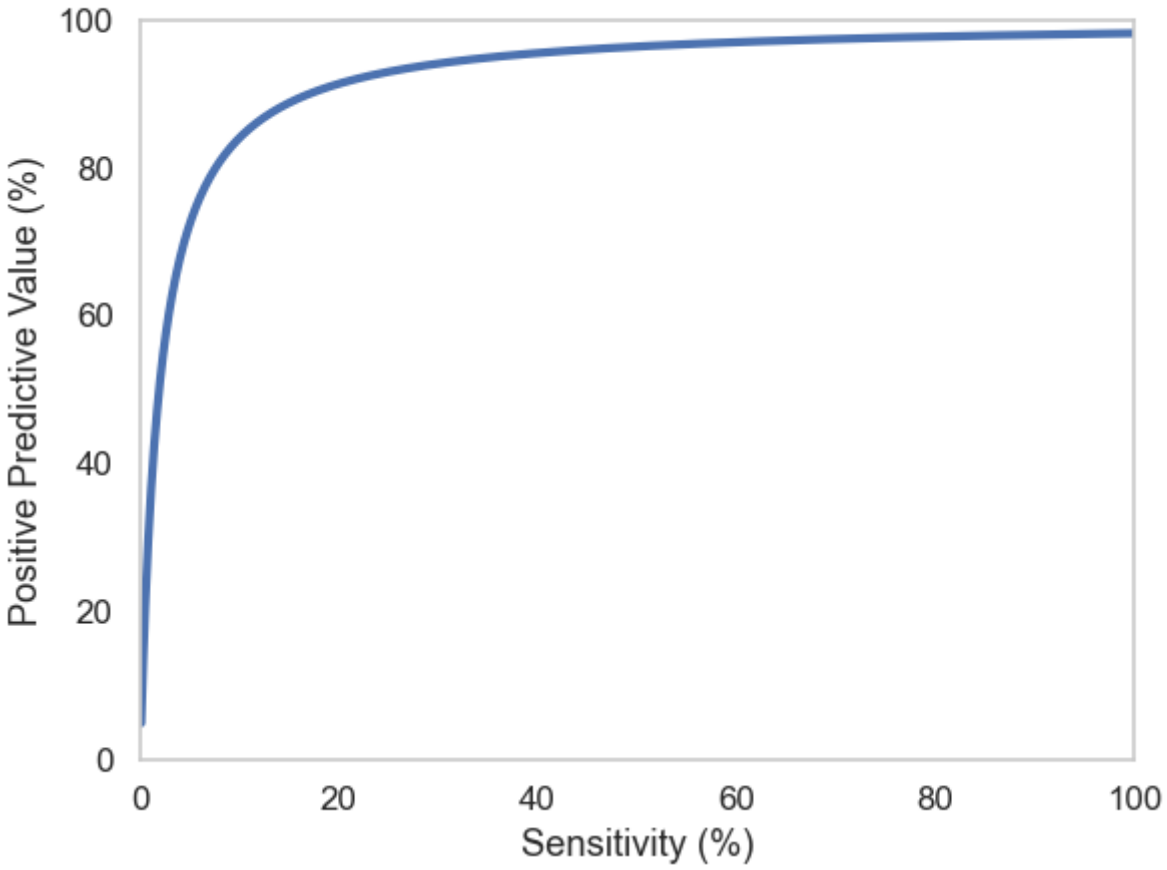


Figure 2: A concave relationship for Sensitivity to PPV can give test developers a false sense of security, regarding their usefulness.

The consequence is that a slight drop in the specificity of your test can have catastrophic effects on your test's usefulness. Even a minor degradation in specificity can mean that many of the test failures are false alarms.

The naive practitioner often attempts to increase a team's test-automation levels by encouraging scenario testing, focusing on checking if a feature is present and 'working'. This soon results in a preponderance of tests that fail intermittently. You now have an app that may be flaky, a bunch of tests that definitely are flaky and no easy route to refactor your way to safety.

In case you're wondering about the effects of Specificity and Sensitivity on Negative Predictive Value. That is the usefulness of the test to show that you are all-clear if you are actually all-clear. You can see in figures 2 & 3 that they remain at relatively high levels in both scenarios

Bonus: The code for these graphs can be found on Github (https://github.com/phoughton/predictive_value).

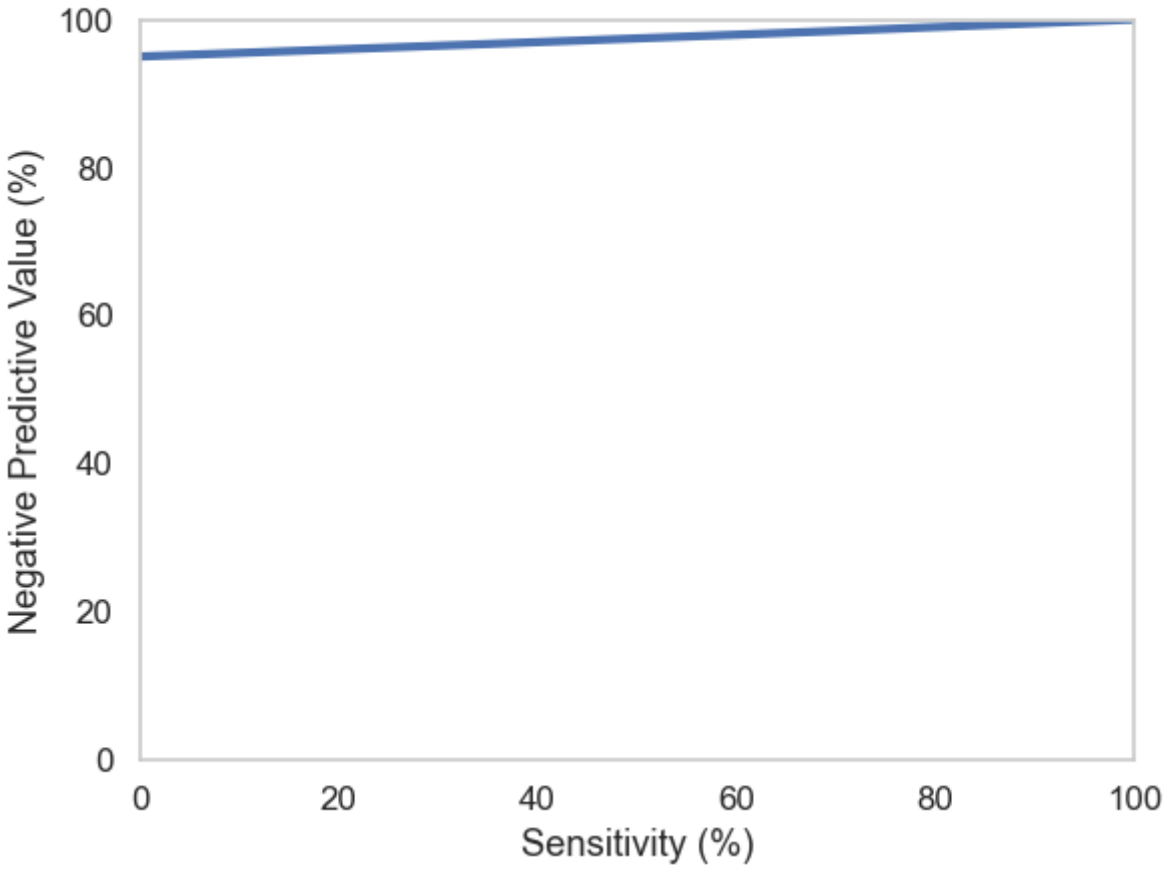
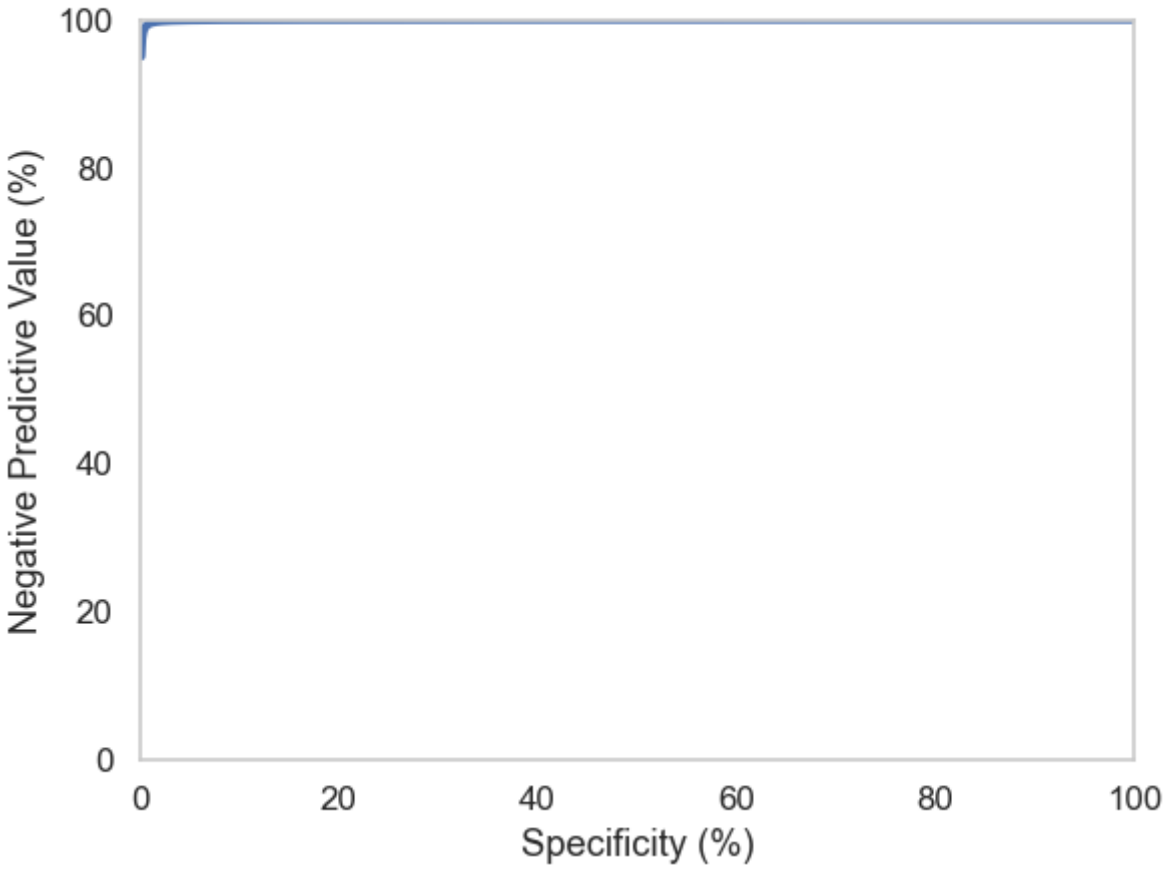


Figure 3: Due to the majority of test runs being on a working system, the varying of Sensitivity has little impact on the Negative Predictive Value of the test.



*| | | — | | Figure 4: Due to the majority of test runs being on a working system, the varying of Specificity also has little impact on the Negative Predictive Value. |

Figure 4: Due to the majority of test runs being on a working system, the varying of Specificity also has little impact on the Negative Predictive Value.*