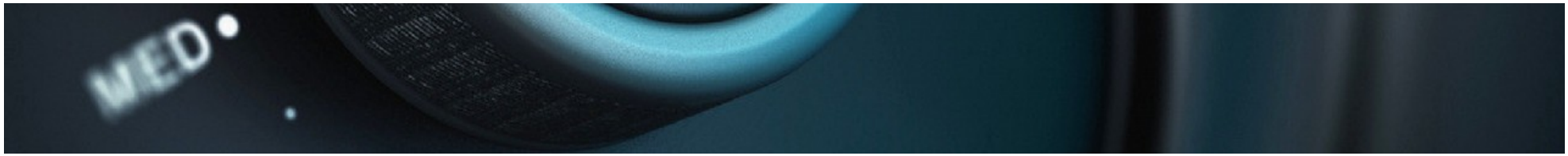
How to perform Quality Assurance for Machine Learning models?

Dhaval M Follow

Nov 27, 2018 · 6 min read



I read it somewhere —

“Don’t worry about the testing a product which includes machine learning algorithms — you just need to make sure that it’s returning accurate results”

- how paradoxical?

Lately, I have faced a dilemma with quantifying quality assurance for ML models. Machine learning and AI is growing rapidly, however, with the added complexity of this field it is a challenge to perform QA. I have observed QA teams completely going in the wrong direction when working on AI projects. Even customers are unclear on how to handle the QA aspects before incorporating AI models within their products.
Some of the questions I have come across are:
- How to perform regression for AI models? (Regression: Regression testing is re-running functional and non-functional tests to ensure that previously developed and tested software still performs after a change)
- How to ensure that the existing model in production does not collapse

when the underlying change is released?
- How to ensure that accuracy is maintained on production?

I will try to answer some of these questions based on my reading and hands-on experience with model building.
But before that some background on Machine learning concepts from QA perspective.

*"Machine Learning is the field of study that gives computers the ability to learn without being **explicitly** programmed."*

If we look at the black box testing approach, any machine learning model is similar to any other algorithm based approach.
There are some inputs which generate some outputs. These outputs can be verified against some expected results.

For white box testing, the philosophy is different and a bit more complex.
The key difference is:
1. Instead of an algorithm, there is a function that is written
2. This function is fitted on by a specific algorithm on given data.

So it is still possible to use the classical approach to test ETL through model coefficients however it is not possible to apply the same philosophy for outputs. On top of that models can behave differently for same input during each run.

So let's break down the overall QA process.

Code Review:
A model review should be performed based on the best practices of model selection for given data. This is similar to finding performing code review and identifying the baseline best practices for coding. Some of the details pertaining to model as shown below help in conformance of the best-suited model.

You will still need a good data scientist to cater on case to case basis. There are two other parameters that can help with identifying the QA accuracy. Bias and Variance.

Bias:

This is the statistical bias or prediction bias of the model. Not bias used in an activation function. Prediction bias is the difference between the expected prediction accuracy of our model and true prediction accuracy.

Variance:

This is the statistical variance of the model. The variance is a measure of the variability of our model's predictions if we repeat the learning process multiple times with small fluctuations in the training set. The more sensitive the model-building process is towards these fluctuations, the higher the variance.

Another observation I had is to use Precision and Recall while evaluating model performance compared to just accuracy. Lets say we are building a spam filter for emails and observe below scenario after filtering 150 emails.
Equation for Accuracy = (TP + TN)/(TP + TN + FP + FN)

TP = True Positive

TN = True Negative

FP = False Positive

FN = False Negative

Execution 1:

For above model Accuracy = 73.3%. This seems to be a decent accuracy. However lets assume that the classifier is dumb and classifies all the emails as not spam i.e. negative

Execution 2:

For above model Accuracy = 83.3%. This is the accuracy paradox.
To avoid this situation we should use Precision or Positive Predictive Value and Recall or Sensitivity.

Precision = TP/(TP + FP)
Recall = TP/(TP + FN)

Ideally Precision = Recall = Should be 1.0

One can even use a combination of both precision and recall through F1 Score as a harmonic mean:

$$F1 = 2^{*}(Precision \ ^{*} Recall)/(Precision + Recall)$$

For a typical QA lets review some scenarios:

1. There is a web service hosted which uses the underlying machine learning model. There is an updated model released.
In this case, it is a sort of black box testing scenario. Load the test data set and verify it has an acceptable output of F1 score. It may not exactly match, however within the acceptable dispersion rate. So you need to set these thresholds beforehand.
2. There is a machine learning function deployed and one needs to verify if the data is processed correctly.
In this case, it is a sort of white box testing scenario. Use unit testing approach to find model values at each stage. Create test cases with these values to mark it as Pass/Fail at each stage.
3. Evaluate the impact of a change in production data and find the ideal scenario to handle it.

In a controlled environment, random seeding may help you to find results for consistent production inputs. However sometimes in production, the same input produces different output. In this case, run the model multiple times. Identify the threshold for F1 score and capture the outliers. If outliers are within the range of acceptable limits, continue the model, otherwise, reject the code for further fine-tuning.

4. A customer wants to run UAT and needs a full regression test cases

Describe the black box to the customer. Provide all the test data and web service to observe the output. Describe the white box testing layers and expected outputs at each level. Create a model quality report (Get this from your data scientist or let me know I can help you with this.)

5. A customer wants to ensure that model in production will not break with upcoming released

Perform black box smoke testing with test data. Run multiple iterations with the new data. Evaluate F1 score and identify if each run has acceptable F1 score and outliers do not cross the threshold.

6. How do I know if the threshold kept for model evaluation is adequate

This has more to do with the data science and essentially the acceptable standard deviation set for the model. Some of the models use class output or probability output. Some of them use a confusion matrix. Some other methods are to use area under the curve. Use data visualizations with respect to these methods to find the error minima.

These are few observations and actions that I have learned from model building exercises. This area is vast and has huge scope of evolution. Do let me know in comments about your experiences.

Dhaval Mandalia enjoys data science, project management, training executives and writes about management strategies. He's also a contributing member in the management association community in Gujarat. Follow him on Twitter and LinkedIn.

Machine Learning Data Science AI Quality Assurance