PROBABILITY CALIBRATION

# Why Calibrators? Part 1 of the Series on Probability Calibration

Why and when should you calibrate the probabilities of a classifier

Jason Yonglin Wu · Following

Published in Towards Data Science · 7 min read · Oct 4, 2020

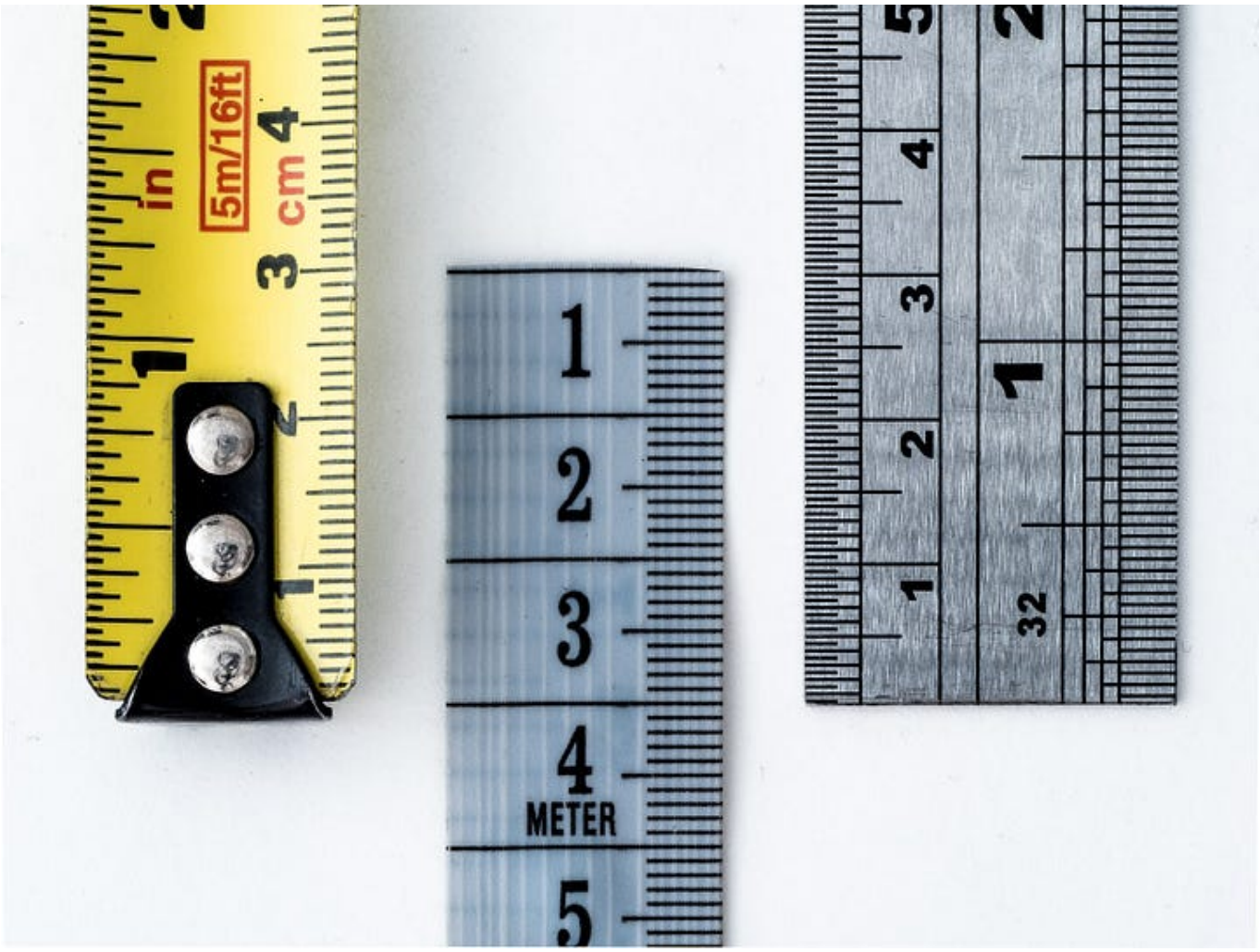



Photo by William Warby on Unsplash

The topic of this series is something that appears sparingly in standard ML courses but frequently in the industry — probability calibration. In the first post of the series, I will give a general introduction to probability calibration for classifiers and discuss when it makes sense to use calibrators.

## Introduction

In machine learning, sometimes **probabilistic** classifiers are needed — classifiers that not only return the most likely class label, but also the probability of that class. A probabilistic classifier is well-calibrated when the predicted probability matches the true probability of the event of interest. For example, if a fraud classifier returns 0.1, or 10%, for the likelihood of a particular credit card application to be fraudulent, this number is

considered well-calibrated if similar types of applications are truly fraudulent on average in 1 of 10 samples.

This is important when the absolute value of the predicted probability, instead of just the rank order, matters for the modeler. An example of this is sports betting.

Before the 2018 World Cup, VegasInsider.com posted 5/1 odds of Germany winning the tournament, which means for every dollar, you get $6 back (original dollar plus $5 payout) if Germany wins and $0 otherwise. So if you want to bet on Germany, you'd better be sure that Germany has more than ⅙ chance of winning. If you built a classifier to predict the winning probability of each team, and its output is Germany: 0.25 and England: 0.1, you want to make sure that classifier is telling you Germany has a 25% chance of winning, instead of merely saying Germany has a better chance than England.


Know your odds, before making your bet. Photo by Kay on Unsplash

## Calibration Plot

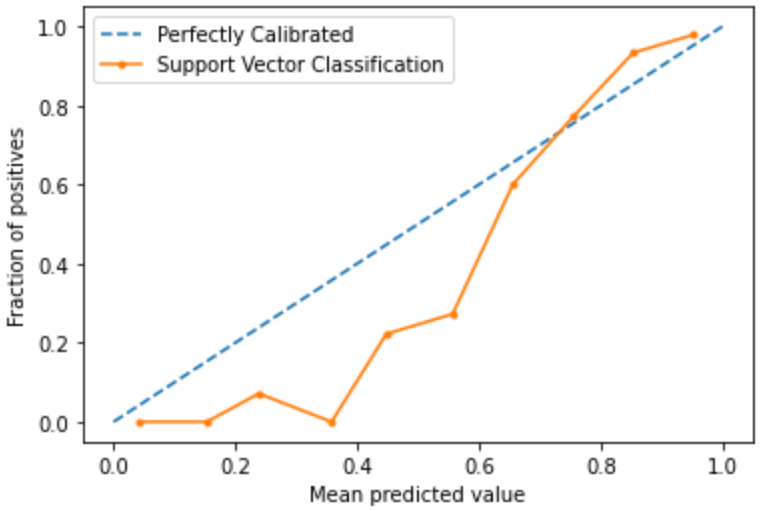
A calibration plot is a standard way to check how calibrated a classifier is on a given set of data with known outcomes. (It only works with binary classifiers; for multi-class classifiers, a separate calibration plot is needed for each class) To create the calibration plot, the following steps are followed.

1. Score the samples in the dataset using the classifier.
2. Bin the data into groups based on their predicted positive class probabilities.
3. Compute the fraction of actual positive in each bin.

4. For each bin, plot a point with fractions of actual positive on the y-axis and the midpoint of the bin (average predicted probability) on the x-axis

Let's make a calibration plot on a real dataset, using sklearn's calibration_cruve

```
from sklearn.datasets import make_classification, load_breast_cancer
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.calibration import calibration_curve
from matplotlib import pyplot

# generate 2 class dataset
X, y = sklearn.datasets.load_breast_cancer(return_X_y=True)

# split into train/test sets
trainX, testX, trainy, testy = train_test_split(X, y, test_size=0.5,
random_state=2)

# fit a model
model = SVC()
model.fit(trainX, trainy)

# predict probabilities
probs = model.decision_function(testX)

# reliability diagram
fop, mpv = calibration_curve(testy, probs, n_bins=10, normalize=True)

# plot perfectly calibrated
pyplot.plot([0, 1], [0, 1], linestyle=' - ', label='perfectly
calibrated')

# plot model reliability
pyplot.plot(mpv, fop, marker='.', label='Support Vector
Classification')
pyplot.xlabel('Mean predicted value')
pyplot.ylabel('Fraction of positives')
pyplot.legend()
pyplot.show()
```




A perfectly calibrated classifier has a calibration curve in the form of y = x, as shown as the blue dotted line in the graph. Comparing the calibration curve for the SVC classifier against the perfect curve, we can see that it predicts a probability that's too low on the lower end, and too high on the higher end. This is typical for maximum-margin methods; a thorough explanation can be found here.

When outputs from out-of-the-box classifiers are not well-calibrated, which is the case in our example above, a calibrator can be trained to correct that. It is a mapping from the raw classifier outputs to the calibrated probabilistic scores. How to train such calibrators is a topic for the next blog post, however, before training the calibrator, we should first ask ourselves whether a calibrator is absolutely necessary. It is also worth noting that in real life, we can never achieve a 'perfectly calibrated' classifier, even with a calibrator. How close we need our calibration curve to match the perfect line is highly dependent on the specific use case.

## Google's Recommendation

Google engineers expressed pretty strong views against calibrators in their Machine Learning Crash Course — Classification: Prediction Bias:

> *You might be tempted to correct prediction bias by post-processing the learned model — that is, by adding a* ***calibration layer*** *that adjusts your model's output to reduce the prediction bias. For example, if your model has +3% bias, you could add a calibration layer that lowers the mean prediction by 3%. However, adding a calibration layer is a bad idea for the following reasons:*
>
> *You're fixing the symptom rather than the cause.*
>
> *You've built a more brittle system that you must now keep up to date.*
>
> *If possible, avoid calibration layers. Projects that use calibration layers tend to become reliant on them — using calibration layers to fix all their model's sins.* ***Ultimately, maintaining the calibration layers can become a nightmare.***

## When to Use Calibrators

In the same article, Google also listed possible root causes of prediction bias:

- Incomplete feature set
- Noisy data set
- Buggy pipeline
- Biased training sample
- Overly strong regularization

I strongly recommend going through that list and trying to fix those issues before using calibrators.

That being said, despite google's well-intentioned warnings, calibrators are used quite often. This does not mean that all of them are created by bad engineers. In practice, things are a lot more complicated and nuanced, and sometimes it's impossible to 'fix the cause rather than the symptom', Here are some typical scenarios in which uses of calibrators are well justified:

## Some types of classifiers are Inherently not well calibrated

Google uses logistic regression as their example and claims *'Logistic regression predictions should be unbiased.'* However, many other types of classifiers, e.g. random forest or SVM, are not. Whether a particular type of classifier is well calibrated depends on its learning algorithm and loss function. For a detailed comparison of how well some common types of classifiers are calibrated and a more in-depth explanation of the underlying reasons, check out scikit-learn's guide on probability calibration.

## Undersampling or Oversampling During Training

These are techniques used in practice to deal with imbalances in the dataset. For example, in credit card transaction datasets, you can expect to have a large number of false (non-fraudulent) outcomes and a relatively smaller number of true outcomes. It is often challenging to train standard ML models on an imbalanced dataset — having few instances of the minority class means that the learning algorithm is often unable to generalize the behavior of the minority class well. This often leads to poor prediction accuracy.

One typical strategy for dealing with this is resampling, either by undersampling — removing samples of the majority class at random, or oversampling — replicating the minority class in the dataset. This strategy improves the prediction accuracy at the expense of introducing differences

in class distributions between the training set and test set. Even if a classifier is well-calibrated on the training set, it is no longer calibrated on the test set since the distribution has changed.

This paper examines this problem in detail and introduces a method to obtain a more calibrated output, which essentially adds a calibration layer between the raw output and final output.

## Label Shift

Label shift is one type of distribution shift in machine learning, typical in production ML systems. In this case, P(X|y) remains the same but P(y) changes. This will cause models to produce uncalibrated results. Using the credit card transaction example again — maybe fraud was a big problem in



to correct the calibration of the model using more recent data and the new base fraud rate. This blogpost examines this phenomenon in detail and proposes a nice solution.

Photo by Chris Lawton on Unsplash

## Partially Observed Dataset

In production ML systems, labels sometimes come from observed events, such as whether users watched a video or repaid their loans, and this means that we only have labels on the dataset with observed events. For example, in credit underwriting, we only observe the payment patterns for the loans that were given out. If we train a model on the training data with observed labels, it will not be well-calibrated because the training data will have way fewer loans than the test dataset, which includes all incoming applications (assuming the current production model does a good job rejecting loan applications with high default probabilities).

This is a well-known feedback loop problem in ML and one way to recover the right data distribution on the test set is using exploration — approving

some loan applications that are typically rejected and upweighting those loans. However, that is an expensive strategy that is undesirable on a much larger training set. Even if money isn't an issue, it will suffer from other problems like high variance and dataset shift. Again, this is a case where it's difficult to train a well-calibrated model out of the box and it's more efficient to train a calibrator on a well-curated testing dataset.

## Summary

In this blog post, I explained why it is important for classifiers to be calibrated, and discussed when it makes sense to use a calibrator to correct the output of classifiers, instead of fixing the algorithm or the training data. In the next few posts, I will cover different methods of training calibrators and ways to evaluate them.

Maching Learning | Probability | Calibration | Probability Calibration

· 8 min read · Oct 20, 2019 | 6 min | May 28, | 3 min read · Ju

View list

## Written by Jason Yonglin Wu

Following

172 Followers · Writer for Towards Data Science

Applied ML Scientist at Affirm, Inc

### More from Jason Yonglin Wu and Towards Data Science


Jason Yonglin Wu in Affirm Tech Blog

## Interpreting Machine Learning Models using Contour Plots

As data scientists, it's custom to prefer models that are interpretable regardless of...

8 min read · Jun 15, 2018

325 2

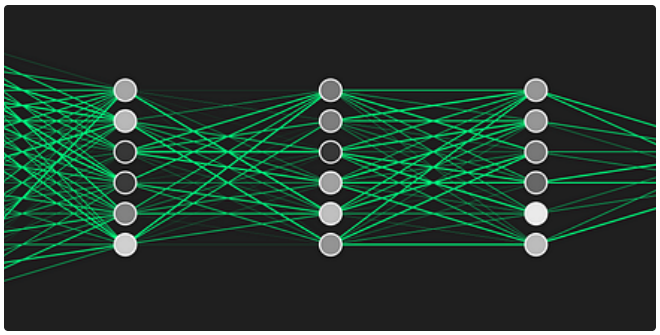

Antonis Makropoulos in Towards Data Science

## How to Build a Multi-GPU System for Deep Learning in 2023

This story provides a guide on how to build a multi-GPU system for deep learning and...

10 min read · Sep 17

552 11



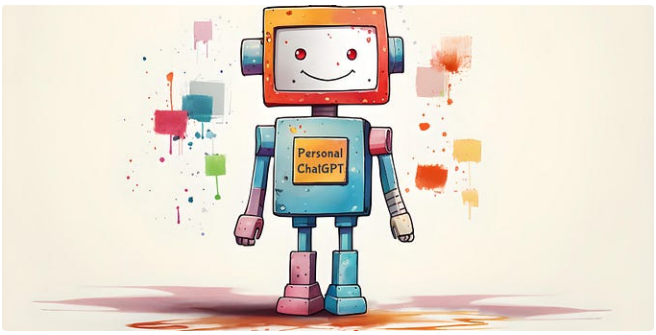
Callum Bruce in Towards Data Science

## How to Program a Neural Network

A step-by-step guide to implementing a neural network from scratch

· 14 min read · Sep 24

485 4




Robert A. Gonsalves in Towards Data Science

## Your Own Personal ChatGPT

How you can fine-tune OpenAI's GPT-3.5 Turbo model to perform new tasks using you...

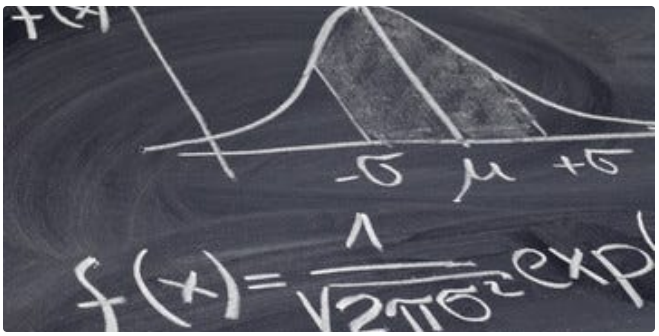
· 15 min read · Sep 8

620 7

See all from Jason Yonglin Wu

See all from Towards Data Science

## Recommended from Medium



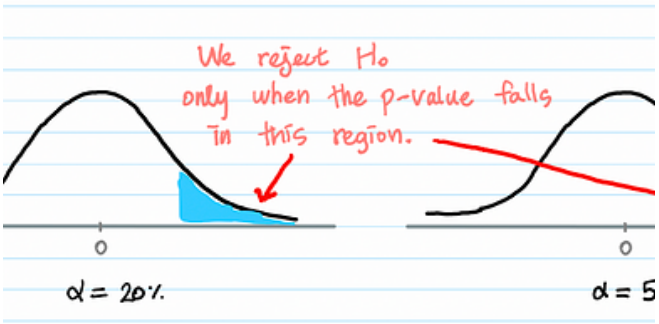

Charlie Lai

### Probability Density Function is Not a Probability

In this article, I'd like to explain why a probability density function(pdf) is not a...

2 min read · Sep 24

32




Ms Aerin in IntuitionMath

### Chi Square Test—Intuition, Examples, and Step-by-Step...

The best way to see if two variables are related.

· 15 min read · Feb 13

407 3

## Lists

**Staff Picks**

469 stories · 338 saves

**Stories to Help You Level-Up at Work**

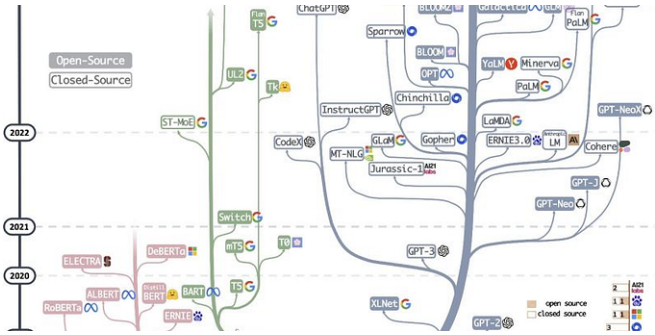
19 stories · 239 saves

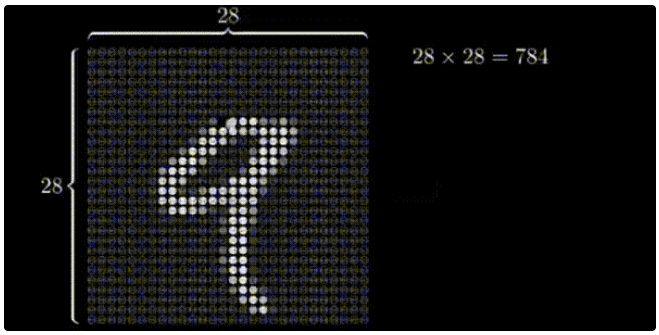
**Self-Improvement 101**

20 stories · 690 saves

**Productivity 101**

20 stories · 625 saves




Haifeng Li

## A Tutorial on LLM

Generative artificial intelligence (GenAI), especially ChatGPT, captures everyone's...

15 min read · Sep 14

805

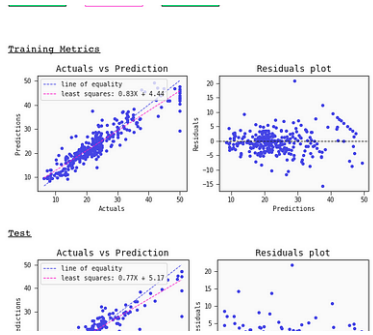
Sadaf Saleem

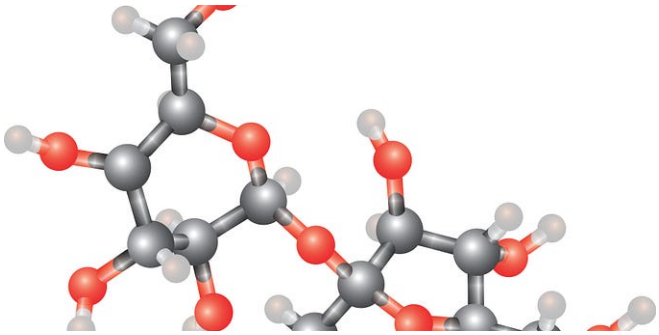
## Neural Networks in 10mins. Simply Explained!

What are Neural Networks?

9 min read · May 15

444 2



Casper Skern Wilstrup

## Symbolic Regression: a Simple and Friendly Introduction

Symbolic Regression is like a treasure hunt for the perfect mathematical equation to...

3 min read · May 5

Eli Hatcher

## Variational Quantum Eigensolvers

What they are, why they're cool, and how they work

5 min read · Nov 9, 2022


See more recommendations