




# calzone: A Python package for measuring calibration of probabilistic models for classification

Kwok Lung Fan<sup>1</sup>, Gene Pennello<sup>1</sup>, Qi Liu<sup>1</sup>, Nicholas Petrick<sup>1</sup>, Ravi K. Samala<sup>1</sup>, Frank W. Samuelson<sup>1</sup>, Yee Lam Elim Thompson<sup>1</sup>, and Qian Cao<sup>1¶</sup>

<sup>1</sup> U.S. Food and Drug Administration ¶ Corresponding author

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

## Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Open Journals](#) 

## Reviewers:

- [@openjournals](#)

Submitted: 01 January 1970

Published: unpublished

## License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#))

## Summary

calzone is a Python package for evaluating the calibration of probabilistic outputs of classifier models. It provides a set of functions and classes for visualizing calibration and computing calibration metrics given a representative dataset with the model's predictions and true class labels. The metrics provided in calzone include: Expected Calibration Error (ECE), Maximum Calibration Error (MCE), Hosmer-Lemeshow (HL) statistic, Integrated Calibration Index (ICI), Spiegelhalter's Z-statistics and Cox's calibration slope/intercept. The package is designed with versatility in mind. For many of the metrics, users can adjust the binning scheme and toggle between top-class or class-wise calculations.

## Statement of need

Classification is one of the most fundamental tasks in machine learning. Classification models are often evaluated by a proper scoring rule, such as cross-entropy or mean square error. Examination of the discrimination performance (resolution), such as AUC or Se/Sp are also used to evaluate the model performance. However, the reliability or calibration performance of the model is often overlooked.

Bröcker (2009) has shown that the proper scoring rule can be decomposed into the resolution and reliability. That means even if the model has high resolution (high AUC), it may not be a reliable or calibrated model. In many high-risk machine learning applications, such as medical diagnosis, the reliability of the model is of paramount importance.

We refer to calibration as the agreement between the predicted probability and the true posterior probability of a class-of-interest,  $P(D = 1 | \hat{p} = p) = p$ . This is also termed as moderate calibration by Calster & Steyerberg (2018).

In the calzone package, we provide a set of functions and classes for calibration visualization and metrics computation. Existing libraries such as scikit-learn are often not dedicated to calibration metrics computation and don't provide calibration metrics computation that are widely used in the statistical literature. Other libraries are focused on implementing calibration methods instead of ways to evaluate calibration [TODO: cite].

## Functionality

### Reliability Diagram

The reliability diagram is a graphical representation of the calibration of a classification model (Bröcker & Smith, 2007). It groups the predicted probabilities into bins and plots the mean

38 predicted probability against the empirical frequency in each bin. The reliability diagram can  
39 be used to assess the calibration of the model and to identify any systematic errors in the  
40 predictions. In addition, calzone gives the option to also plot the confidence interval of the  
41 empirical frequency in each bin. The confidence intervals are calculated using Wilson's score  
42 interval (Wilson, 1927). We provide an example analysis in the example\_data folder using  
43 beta-binomial distribution (Griffiths, 1973). Users can generate simulated data using the  
44 fake\_binary\_data\_generator class in the utils module.

```
from calzone.utils import reliability_diagram
from calzone.vis import plot_reliability_diagram

wellcal_data_loader = data_loader(
    data_path="example_data/simulated_welldata.csv"
)

reliability, confidence, bin_edges, bin_counts = reliability_diagram(
    wellcal_data_loader.labels,
    wellcal_data_loader.probs,
    num_bins=15,
    class_to_plot=1
)

plot_reliability_diagram(
    reliability,
    confidence,
    bin_counts,
    error_bar=True,
    title='Class 1 reliability diagram for well calibrated data'
)
```

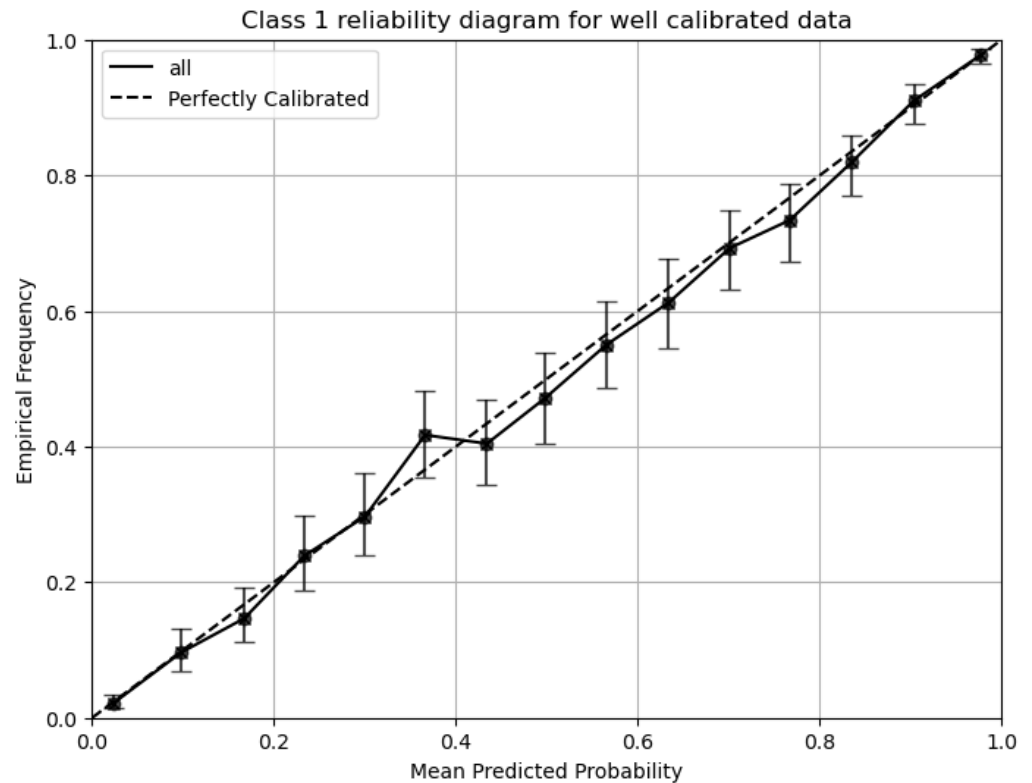


Figure 1: Reliability Diagram for well calibrated data

## Calibration metrics

calzone provides functions to compute various calibration metrics. The `CalibrationMetrics()` class allows the user to compute the calibration metrics in a more convenient way. The following are metrics that are currently supported in calzone:

### Expected Calibration Error (ECE) and Maximum Calibration Error (MCE)

Expected Calibration Error (ECE), Maximum Calibration Error (MCE) and other binning-based methods (Guo et al., 2017; Pakdaman Naeini et al., 2015) aim to measure the average deviation between predicted probability and true probability. We provide the option to use equal-width binning or equal-count binning, labeled as ECE-H and ECE-C respectively. Users can also choose to compute the metrics for the class-of-interest or the top-class. In the case of class-of-interest, calzone will evaluate the calibration of a one-vs-rest classification problem. The following snippet demonstrates how these metrics are calculated in our package:

```
from calzone.metrics import calculate_ece_mce

reliability, confidence, bin_edges, bin_counts = reliability_diagram(
    wellcal_dataloader.labels,
    wellcal_dataloader.probs,
    num_bins=10,
    class_to_plot=1,
    is_equal_freq=False
)

ece_h_classone, mce_h_classone = calculate_ece_mce(
```

```

        reliability,
        confidence,
        bin_counts=bin_counts
    )

```

## 57 Hosmer-Lemeshow statistic (HL)

58 The Hosmer-Lemeshow (HL) statistical test is for evaluating the calibration of a probabilistic  
 59 model. It is a chi-square-based test that compares the observed and expected number of  
 60 events in each bin. The null hypothesis is that the model is well calibrated. HL-test first bins  
 61 data into predicted probability bins (equal-width  $H$  or equal-count  $C$ ) and the test statistic is  
 62 calculated as:

$$HL = \sum_{m=1}^M \frac{(O_{1,m} - E_{1,m})^2}{E_{1,m}(1 - \frac{E_{1,m}}{N_m})} \sim \chi_{M-2}^2$$

63 where  $E_{1,m}$  is the expected number of class-of-interest events in the  $m^{th}$  bin,  $O_{1,m}$  is the  
 64 observed number of class-of-interest events in the  $m^{th}$  bin,  $N_m$  is the total number of  
 65 observations in the  $m^{th}$  bin, and  $M$  is the number of bins. In calzone, the HL-test can be  
 66 computed as follows:

```

from calzone.metrics import hosmer_lemeshow_test

HL_H_ts, HL_H_p, df = hosmer_lemeshow_test(
    reliability,
    confidence,
    bin_count=bin_counts
)

```

67 In calzone, user can sepecify the degree of freedom of the HL test by setting the df para-  
 68 meter. This is useful because when performing the HL test on validation sets that are not  
 69 used in training, the degree of freedom of the HL test changes from  $M - 2$  to  $M$  [TODO:  
 70 cite]. Intuitively,  $\frac{(O_{1,m} - E_{1,m})^2}{E_{1,m}(1 - \frac{E_{1,m}}{N_m})}$  is the difference squared divided by the variance of a binomial  
 71 distribution and follows a chi-square distribution with 1 degree of freedom. Hence, the sum  
 72 of  $M$  chi-square distributions with 1 degree of freedom is a chi-square distribution with  $M$   
 73 degrees of freedom if the data has no effect on the model.

## 74 Cox's calibration slope/intercept

75 Cox's calibration slope/intercept is a non-parametric method for assessing the calibration of a  
 76 probabilistic model (COX, 1958). A new logistic regression model is fitted to the data, with  
 77 the predicted odds ( $\frac{p}{1-p}$ ) as the dependent variable and the true probability as the independent  
 78 variable. The slope and intercept of the regression line are then used to assess the calibration  
 79 of the model. A slope of 1 and intercept of 0 indicates perfect calibration. To test whether  
 80 the model is calibrated, fix the slope to 1 and fit the intercept. If the intercept is significantly  
 81 different from 0, the model is not calibrated. Then, fix the intercept to 0 and fit the slope.  
 82 If the slope is significantly different from 1, the model is not calibrated. In calzone, Cox's  
 83 calibration slope/intercept can be computed as follows:

```

from calzone.metrics import cox_regression_analysis

cox_slope, cox_intercept, cox_slope_ci, cox_intercept_ci = cox_regression_analysis(
    wellcal_data_loader.labels,
    wellcal_data_loader.probs,
    class_to_calculate=1,
    print_results=True,
)

```

```
fix_slope=True
)
```

84 The values of the slope and intercept can represent miscalibration throughout the range of  
 85 probability outputs. The integrated calibration index (ICI) is very similar to Expected calibration  
 86 error (ECE). It also tries to measure the average deviation between predicted probability and  
 87 true probability. However, ICI does not use binning to estimate the true probability of a group  
 88 of samples with similar predicted probability. Instead, ICI uses curve smoothing techniques  
 89 to fit the regression curve and uses the regression result as the true probability (Austin &  
 90 Steyerberg, 2019). The ICI is then calculated using the following formula:

$$ICI = \frac{1}{n} \sum_{i=1}^n |f(p_i) - p_i|$$

91 where  $f$  is the fitting function and  $p$  is the predicted probability. The curve fitting is usually  
 92 done with loess regression. However, it is possible to use any curve fitting method to calculate  
 93 the ICI. In calzone, we provide Cox's ICI and loess ICI support while the user can also use any  
 94 curve fitting method to calculate the ICI using functions in calzone.

```
from calzone.metrics import (
    cox_regression_analysis,
    lowess_regression_analysis,
    cal_ICI_cox
)
```

```
### calculating cox ICI
```

```
cox_ici = cal_ICI_cox(
    cox_slope,
    cox_intercept,
    wellcal_data_loader.probs,
    class_to_calculate=1
)
```

```
### calculating loess ICI
```

```
loess_ici, loess_fit_p, loess_fit_p_correct = lowess_regression_analysis(
    wellcal_data_loader.labels,
    wellcal_data_loader.probs,
    class_to_calculate=1,
    span=0.5,
    delta=0.001,
    it=0
)
```

95 Notice that flexible curve fitting methods such as Loess regression are very sensitive to the  
 96 choice of span and delta parameters. The user can visualize the fitting result to avoid overfitting  
 97 or underfitting.

## 98 Spiegelhalter's Z-test

99 Spiegelhalter's Z-test is a test of calibration proposed by Spiegelhalter in 1986 (Spiegelhalter,  
 100 1986). It uses the fact that the Brier score can be decomposed into:

$$B = \frac{1}{N} \sum_{i=1}^N (x_i - p_i)^2 = \frac{1}{N} \sum_{i=1}^N (x_i - p_i)(1 - 2p_i) + \frac{1}{N} \sum_{i=1}^N p_i(1 - p_i)$$

101 And the TS of Z test is defined as:

$$Z = \frac{B - E(B)}{\sqrt{\text{Var}(B)}} = \frac{\sum_{i=1}^N (x_i - p_i)(1 - 2p_i)}{\sum_{i=1}^N (1 - 2p_i)^2 p_i(1 - p_i)}$$

102 and it is asymptotically distributed as a standard normal distribution. In calzone, it can be  
103 calculated using:

```
from calzone.metrics import spiegelhalter_z_test

z, p_value = spiegelhalter_z_test(
    wellcal_dataloader.labels,
    wellcal_dataloader.probs,
    class_to_calculate=1
)
```

#### 104 Metrics class

105 calzone also provides a class called CalibrationMetrics() to calculate all the metrics men-  
106 tioned above. The user can also use this class to calculate the metrics.

```
from calzone.metrics import CalibrationMetrics

metrics = CalibrationMetrics(class_to_calculate=1)

CalibrationMetrics.calculate_metrics(
    wellcal_dataloader.labels,
    wellcal_dataloader.probs,
    metrics='all'
)
```

#### 107 Other features

##### 108 Confidence intervals

109 In addition to point estimates of calibration performance, calzone also provides bootstrapping  
110 to calculate the confidence intervals of the metrics. The user can specify the number of  
111 bootstrap samples and the confidence level.

```
from calzone.metrics import CalibrationMetrics

metrics = CalibrationMetrics(class_to_calculate=1)

CalibrationMetrics.bootstrap(
    wellcal_dataloader.labels,
    wellcal_dataloader.probs,
    metrics='all',
    n_samples=1000
)
```

112 and a structured numpy array will be returned.

##### 113 Subgroup analysis

114 calzone will perform subgroup analysis by default in the command line user interface. If the  
115 user input CSV file contains a subgroup column, the program will compute metrics for the  
116 entire dataset and for each subgroup.

##### 117 Prevalence adjustment

118 calzone also provides prevalence adjustment to account for prevalence changes between  
119 training data and testing data. Since calibration is defined using posterior probability, a

mere shift in the prevalence of the testing data will result in miscalibration. It can be fixed by searching for the optimal derived original prevalence such that the adjusted probability minimizes a proper scoring rule such as cross-entropy loss. The formula of prevalence adjusted probability is:

$$P'(D = 1|\hat{p} = p) = \frac{\eta'/(1 - \eta')}{(1/p - 1)(\eta/(1 - \eta))} = p'$$

where  $\eta$  is the prevalence of the testing data,  $\eta'$  is the prevalence of the training data, and  $p$  is the predicted probability (Chen et al., 2018; Gu & Pepe, 2010; Horsch et al., 2008; Tian et al., 2020). We search for the optimal  $\eta'$  that minimizes the cross-entropy loss.

## Multiclass extension

calzone also provides multiclass extension to calculate the metrics for multiclass classification. The user can specify the class to calculate the metrics using a 1-vs-rest approach and test the calibration of each class. Alternatively, the user can transform the data and make the problem become a top-class calibration problem. The top-class calibration has a similar format to binary classification, but the class 0 probability is defined as 1 minus the probability of the class with the highest probability, and the class 1 probability is defined as the probability of the class with the highest probability. The labels are transformed into whether the predicted class equals the true class, 0 if not and 1 if yes. Notice that the interpretation of some metrics may change in the top-class transformation.

## Command line interface

calzone also provides a command line interface to calculate the metrics. The user can visualize the calibration curve, calculate the metrics and their confidence intervals using the command line interface. To use the command line interface, the user can run `python cal_metrics.py -h` to see the help message.

## Acknowledgements

The authors acknowledge the Research Participation Program at the Center for Devices and Radiological Health administered by the Oak Ridge Institute for Science and Education through an interagency agreement between the U.S. Department of Energy and the U.S. Food and Drug Administration (FDA).

## Conflicts of interest

The authors declare no conflicts of interest.

## References

- Austin, P. C., & Steyerberg, E. W. (2019). The integrated calibration index (ICI) and related metrics for quantifying the calibration of logistic regression models. *Statistics in Medicine*, 38(21), 4051–4065. <https://doi.org/10.1002/sim.8281>
- Bröcker, J. (2009). Reliability, sufficiency, and the decomposition of proper scores. *Quarterly Journal of the Royal Meteorological Society*, 135(643), 1512–1519. <https://doi.org/10.1002/qj.456>
- Bröcker, J., & Smith, L. A. (2007). Increasing the reliability of reliability diagrams. *Weather and Forecasting*, 22(3), 651–661. <https://doi.org/10.1175/WAF993.1>

- 158 Calster, B. V., & Steyerberg, E. W. (2018). Calibration of prognostic risk scores. In  
159 *Wiley StatsRef: Statistics reference online* (pp. 1–10). John Wiley & Sons, Ltd.  
160 [https://doi.org/https://doi.org/10.1002/9781118445112.stat08078](https://doi.org/10.1002/9781118445112.stat08078)
- 161 Chen, W., Sahiner, B., Samuelson, F., Pezeshk, A., & Petrick, N. (2018). Calibration of  
162 medical diagnostic classifier scores to the probability of disease. *Statistical Methods in*  
163 *Medical Research*, 27(5), 1394–1409. <https://doi.org/10.1177/0962280216661371>
- 164 COX, D. R. (1958). Two further applications of a model for binary regression. *Biometrika*,  
165 45(3-4), 562–565. <https://doi.org/10.1093/biomet/45.3-4.562>
- 166 Griffiths, D. A. (1973). Maximum likelihood estimation for the beta-binomial distribution  
167 and an application to the household distribution of the total number of cases of a disease.  
168 *Biometrics*, 29(4), 637–648. <http://www.jstor.org/stable/2529131>
- 169 Gu, W., & Pepe, M. S. (2010). Estimating the diagnostic likelihood ratio of a continuous  
170 marker. *Biostatistics*, 12(1), 87–101. <https://doi.org/10.1093/biostatistics/kxq045>
- 171 Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural  
172 networks. In D. Precup & Y. W. Teh (Eds.), *Proceedings of the 34th international*  
173 *conference on machine learning* (Vol. 70, pp. 1321–1330). PMLR. <https://proceedings.mlr.press/v70/guo17a.html>
- 174
- 175 Horsch, K., Giger, M. L., & Metz, C. E. (2008). Prevalence scaling: Applications to an  
176 intelligent workstation for the diagnosis of breast cancer. *Academic Radiology*, 15(11),  
177 1446–1457. [https://doi.org/https://doi.org/10.1016/j.acra.2008.04.022](https://doi.org/10.1016/j.acra.2008.04.022)
- 178 Pakdaman Naeini, M., Cooper, G., & Hauskrecht, M. (2015). Obtaining well calibrated  
179 probabilities using bayesian binning. *Proceedings of the AAAI Conference on Artificial*  
180 *Intelligence*, 29(1). <https://doi.org/10.1609/aaai.v29i1.9602>
- 181 Spiegelhalter, D. J. (1986). Probabilistic prediction in patient management and clinical trials.  
182 *Statistics in Medicine*, 5(5), 421–433.
- 183 Tian, J., Liu, Y.-C., Glaser, N., Hsu, Y.-C., & Kira, Z. (2020). Posterior re-calibration  
184 for imbalanced datasets. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan,  
185 & H. Lin (Eds.), *Advances in neural information processing systems* (Vol. 33, pp.  
186 8101–8113). Curran Associates, Inc. [https://proceedings.neurips.cc/paper\\_files/paper/2020/file/5ca359ab1e9e3b9c478459944a2d9ca5-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2020/file/5ca359ab1e9e3b9c478459944a2d9ca5-Paper.pdf)
- 187
- 188 Wilson, E. B. (1927). Probable inference, the law of succession, and statistical inference.  
189 *Journal of the American Statistical Association*, 22(158), 209–212.