




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

Kwok Lung Fan¹, Gene Pennello¹, Qi Liu¹, Nicholas Petrick¹, Ravi K. Samala¹, Frank W. Samuelson¹, Yee Lam Elim Thompson¹, and Qian Cao^{1¶}

¹ 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Published: unpublished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 for visualizing calibration and computation of 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 common applications in machine learning. Classification models are often evaluated by a proper scoring rule - a scoring function that assigns the best score when predicted probabilities match the true probabilities - such as cross-entropy or mean square error ([Gneiting & Raftery, 2007](#)). 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.

Diamond ([1992](#)) had shown that the resolution performance of a model does not indicate the reliability of the model. Bröcker ([2009](#)) later has shown that any proper scoring rule can be decomposed into the resolution and reliability. Thus 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 define 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 has been defined as moderate calibration by Van Calster & Steyerberg ([2018](#)).

In the calzone package, we provide a set of functions and classes for visualizing calibration and evaluating calibration metrics. Existing libraries such as scikit-learn are often not dedicated to calibration metrics computation or lacks calibration metrics that are widely used in the statistical literature. Other libraries such as uncertainty-toolbox are focused on implementing calibration methods instead of calibration assessment. ([Chung et al., 2021](#)).

36 Functionality

37 Reliability Diagram

38 The reliability diagram (also referred to as the calibration plot) is a graphical representation of
39 the calibration of a classification model (Bröcker & Smith, 2007; Murphy & Winkler, 1977). It
40 groups the predicted probabilities into bins and plots the mean predicted probability against the
41 empirical frequency in each bin. The reliability diagram can be used to assess the calibration
42 of the model and to identify any systematic errors in the predictions. In addition, calzone
43 gives the option to also plot the confidence interval of the empirical frequency in each bin.
44 The confidence intervals are calculated using Wilson's score interval (Wilson, 1927). We
45 provide example data in the example_data folder which are simulated using a beta-binomial
46 distribution (Griffiths, 1973). The predicted probabilities are sampled from a beta distribution
47 and the true labels are assigned by performing Bernoulli trials with the sampled probabilities.
48 Users can generate simulated data using the fake_binary_data_generator class in the utils
49 module.

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

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

reliability, confidence, bin_edges, bin_counts = reliability_diagram(
    wellcal_dataloader.labels,
    wellcal_dataloader.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'
)
```

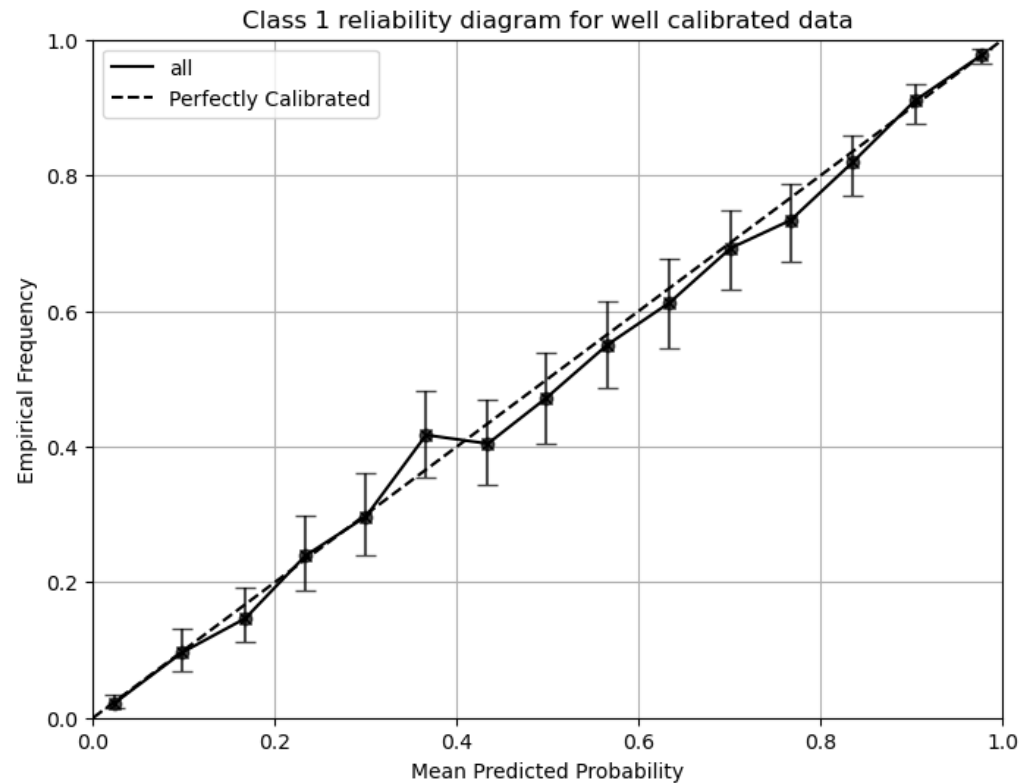


Figure 1: Reliability Diagram for well calibrated data

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 absolute 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
)

```

62 Hosmer-Lemeshow statistic (HL)

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

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

68 where $E_{1,m}$ is the expected number of class-of-interest events in the m^{th} bin, $O_{1,m}$ is the
69 observed number of class-of-interest events in the m^{th} bin, N_m is the total number of
70 observations in the m^{th} bin, and M is the number of bins. In calzone, the HL-test can be
71 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
)

```

72 When performing the HL test on validation sets that are not used in training, the degree
73 of freedom of the HL test changes from $M - 2$ to M (Hosmer Jr et al., 2013). Intuitively,
74 $\frac{(O_{1,m} - E_{1,m})^2}{E_{1,m}(1 - \frac{E_{1,m}}{N_m})}$ is the difference squared divided by the variance of a binomial distribution and
75 follows a chi-square distribution with 1 degree of freedom. Hence, the sum of M chi-square
76 distributions with 1 degree of freedom is a chi-square distribution with M degrees of freedom if
77 the data has no effect on the model. The increase in degree of freedom for validation samples
78 has often been overlooked but it is crucial for the test to maintain the correct type 1 error
79 rate. In calzone, users can specify the degree of freedom of the HL test by setting the df
80 parameter.

81 Cox's calibration slope/intercept

82 Cox's calibration slope/intercept is a regression analysis method for assessing the calibration of
83 a probabilistic model (COX, 1958), which doesn't require binning. A new logistic regression
84 model is fitted to the data, with the predicted odds ($\frac{p}{1-p}$) as the independent variable and the
85 outcome as the dependent variable. The slope and intercept of the regression line are then
86 used to assess the calibration of the model. A slope of 1 and intercept of 0 indicates perfect
87 calibration. To test whether the model is calibrated, fix the slope to 1 and fit the intercept.
88 If the intercept is significantly different from 0, the model is not calibrated. Then, fix the
89 intercept to 0 and fit the slope. If the slope is significantly different from 1, the model is not
90 calibrated. Alternatively, the slope and intercept can be fitted and tested simultaneously using
91 bivariate distribution (McCullagh & Nelder, 1989). This feature is not provided in calzone but
92 user can extract the covariance matrix by printing the result and perform the test manually. In
93 calzone, Cox's 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_dataloader.labels,  
wellcal_dataloader.probs,  
class_to_calculate=1,  
print_results=True,  
fix_slope=True  
)
```

94 The slope and intercept values indicate the type of miscalibration. A slope > 1 shows overcon-
95 fidence at high probabilities and underconfidence at low probabilities (and vice versa). In other
96 word, a slope < 1 (> 1) indicates that the spread of the predicted risks is too large (small)
97 relative to the true risks. A positive intercept indicates general overconfidence (and vice versa).
98 However, even with ideal slope and intercept values, the model may still be miscalibrated due
99 to non-linear effects that Cox's analysis cannot detect.

100 Integrated calibration index (ICI)

101 The integrated calibration index (ICI) is very similar to Expected calibration error (ECE). It
102 also tries to measure the average deviation between predicted probability and true probability.
103 However, ICI does not use binning to estimate the true probability of a group of samples with
104 similar predicted probability. Instead, ICI uses curve smoothing techniques to fit the regression
105 curve and uses the regression result as the true probability (Austin & Steyerberg, 2019). The
106 ICI is then calculated using the following formula:

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

107 where f is the fitting function and p is the predicted probability. The curve fitting is usually
108 done with Locally Weighted Scatterplot Smoothing (LOWESS). However, it is possible to
109 use any curve fitting method to calculate the ICI. In calzone, we provide Cox's ICI and loess
110 ICI support while the user can also use any curve fitting method to calculate the ICI using
111 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_dataloader.probs,  
    class_to_calculate=1  
)
```

```
### calculating LOWESS ICI  
loess_ici, lowess_fit_p, lowess_fit_p_correct = lowess_regression_analysis(  
    wellcal_dataloader.labels,  
    wellcal_dataloader.probs,  
    class_to_calculate=1,  
    span=0.5,  
    delta=0.001,  
    it=0  
)
```

112 Notice that flexible curve fitting methods such as LOWESS regression are very sensitive to

the choice of span and delta parameters. The user can visualize the fitting result to avoid overfitting or underfitting.

Spiegelhalter's Z-test

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

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

And the TS of Z test is defined as:

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

and it is asymptotically distributed as a standard normal distribution. In calzone, it can be 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
)
```

Metrics class

calzone also provides a class called CalibrationMetrics() to calculate all the metrics mentioned 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'
)
```

Other features

Confidence intervals

In addition to point estimates of calibration performance, calzone also provides functionality to compute confidence intervals for all metrics. For most metrics, this is computed through bootstrapping. The user can specify the number of 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
    )

```

and a structured NumPy array will be returned.

Subgroup analysis

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

Prevalence adjustment

calzone also provides prevalence adjustment to account for prevalence changes between training data and testing data. Since calibration is defined using posterior probability, a mere shift in the disease 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 η is the prevalence of the testing data, η' 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 η' that minimizes the cross-entropy loss. The user can specify also specify η' and adjust the probability output directly if that is available.

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.

Validation of methods

We compared the results calculated by calzone with external packages for some metrics to ensure the correctness of the implementation. For reliability diagram, we compared the result with the `sklearn.calibration.calibration_curve()` function in scikit-learn (Pedregosa et al., 2011). For top-class ECE and Spiegelhalter's Z score, we compared the result with the `MAPIE` package (Taquet et al., 2022). For Hosmer-Lemeshow statistic, we compared the result with the `ResourceSelection` package in R language (Lele et al., 2024). Their results are consistent with ours. For other metrics such as ICI, no external package is available, so we compared the result with ECE as they are similar and we obtained reasonably similar results. We include the validation codes in our documentation.

Command line interface

calzone also provides a command line interface. Users can visualize the calibration curve, calculate calibration metrics and their confidence intervals using the command line interface.

169 To use the command line interface, the user can run `python cal_metrics.py -h` to see the
170 help message.

171 Acknowledgements

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

176 Conflicts of interest

177 The authors declare no conflicts of interest.

178 References

- 179 Austin, P. C., & Steyerberg, E. W. (2019). The integrated calibration index (ICI) and related
180 metrics for quantifying the calibration of logistic regression models. *Statistics in Medicine*,
181 38(21), 4051–4065. <https://doi.org/https://doi.org/10.1002/sim.8281>
- 182 Bröcker, J. (2009). Reliability, sufficiency, and the decomposition of proper scores. *Quarterly*
183 *Journal of the Royal Meteorological Society*, 135(643), 1512–1519. <https://doi.org/https://doi.org/10.1002/qj.456>
- 184 Bröcker, J., & Smith, L. A. (2007). Increasing the reliability of reliability diagrams. *Weather*
185 *and Forecasting*, 22(3), 651–661. <https://doi.org/10.1175/WAF993.1>
- 186 Chen, W., Sahiner, B., Samuelson, F., Pezeshk, A., & Petrick, N. (2018). Calibration of
187 medical diagnostic classifier scores to the probability of disease. *Statistical Methods in*
188 *Medical Research*, 27(5), 1394–1409. <https://doi.org/10.1177/0962280216661371>
- 189 Chung, Y., Char, I., Guo, H., Schneider, J., & Neiswanger, W. (2021). Uncertainty toolbox:
190 An open-source library for assessing, visualizing, and improving uncertainty quantification.
191 *arXiv Preprint arXiv:2109.10254*.
- 192 COX, D. R. (1958). Two further applications of a model for binary regression. *Biometrika*,
193 45(3-4), 562–565. <https://doi.org/10.1093/biomet/45.3-4.562>
- 194 Diamond, G. A. (1992). What price perfection? Calibration and discrimination of clinical
195 prediction models. *Journal of Clinical Epidemiology*, 45(1), 85–89. [https://doi.org/https://doi.org/10.1016/0895-4356\(92\)90192-P](https://doi.org/https://doi.org/10.1016/0895-4356(92)90192-P)
- 196 Gneiting, T., & Raftery, A. E. (2007). Strictly proper scoring rules, prediction, and estimation.
197 *Journal of the American Statistical Association*, 102(477), 359–378.
- 198 Griffiths, D. A. (1973). Maximum likelihood estimation for the beta-binomial distribution
199 and an application to the household distribution of the total number of cases of a disease.
200 *Biometrics*, 29(4), 637–648. <http://www.jstor.org/stable/2529131>
- 201 Gu, W., & Pepe, M. S. (2010). Estimating the diagnostic likelihood ratio of a continuous
202 marker. *Biostatistics*, 12(1), 87–101. <https://doi.org/10.1093/biostatistics/kxq045>
- 203 Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural
204 networks. In D. Precup & Y. W. Teh (Eds.), *Proceedings of the 34th international*
205 *conference on machine learning* (Vol. 70, pp. 1321–1330). PMLR. <https://proceedings.mlr.press/v70/guo17a.html>
- 206 Horsch, K., Giger, M. L., & Metz, C. E. (2008). Prevalence scaling: Applications to an

- intelligent workstation for the diagnosis of breast cancer. *Academic Radiology*, 15(11), 1446–1457. <https://doi.org/https://doi.org/10.1016/j.acra.2008.04.022>
- Hosmer Jr, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). *Applied logistic regression*. John Wiley & Sons.
- Lele, S. R., Keim, J. L., & Solymos, P. (2024). *ResourceSelection: Resource selection (probability) functions for use-availability data*. <https://github.com/psolymos/ResourceSelection>
- McCullagh, P., & Nelder, J. A. (1989). *Generalized linear models*. Chapman & Hall / CRC.
- Murphy, A. H., & Winkler, R. L. (1977). Reliability of subjective probability forecasts of precipitation and temperature. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 26(1), 41–47. <http://www.jstor.org/stable/2346866>
- Pakdaman Naeini, M., Cooper, G., & Hauskrecht, M. (2015). Obtaining well calibrated probabilities using bayesian binning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1). <https://doi.org/10.1609/aaai.v29i1.9602>
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., & others. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct), 2825–2830.
- Spiegelhalter, D. J. (1986). Probabilistic prediction in patient management and clinical trials. *Statistics in Medicine*, 5(5), 421–433.
- Taquet, V., Blot, V., Morzadec, T., Lacombe, L., & Brunel, N. (2022). MAPIE: An open-source library for distribution-free uncertainty quantification. *arXiv Preprint arXiv:2207.12274*.
- Tian, J., Liu, Y.-C., Glaser, N., Hsu, Y.-C., & Kira, Z. (2020). Posterior re-calibration for imbalanced datasets. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems* (Vol. 33, pp. 8101–8113). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2020/file/5ca359ab1e9e3b9c478459944a2d9ca5-Paper.pdf
- Van Calster, B., & Steyerberg, E. W. (2018). Calibration of prognostic risk scores. In *Wiley StatsRef: Statistics reference online* (pp. 1–10). John Wiley & Sons, Ltd. <https://doi.org/https://doi.org/10.1002/9781118445112.stat08078>
- Wilson, E. B. (1927). Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158), 209–212.