

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 Cao¹¶

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Published: unpublished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 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.

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 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 such as uncertainty-toolbox are focused on implementing calibration methods and visualization instead of ways to evaluate calibration (Chung et al., 2021).

37 **Functionality**

38 **Reliability Diagram**

39 The reliability diagram is a graphical representation of the calibration of a classification model
40 (Bröcker & Smith, 2007). It groups the predicted probabilities into bins and plots the mean
41 predicted probability against the empirical frequency in each bin. The reliability diagram can
42 be used to assess the calibration of the model and to identify any systematic errors in the
43 predictions. In addition, calzone gives the option to also plot the confidence interval of the
44 empirical frequency in each bin. The confidence intervals are calculated using Wilson's score
45 interval (Wilson, 1927). We provide an example analysis in the example_data folder using
46 beta-binomial distribution (Griffiths, 1973). Users can generate simulated data using the
47 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'
)
```

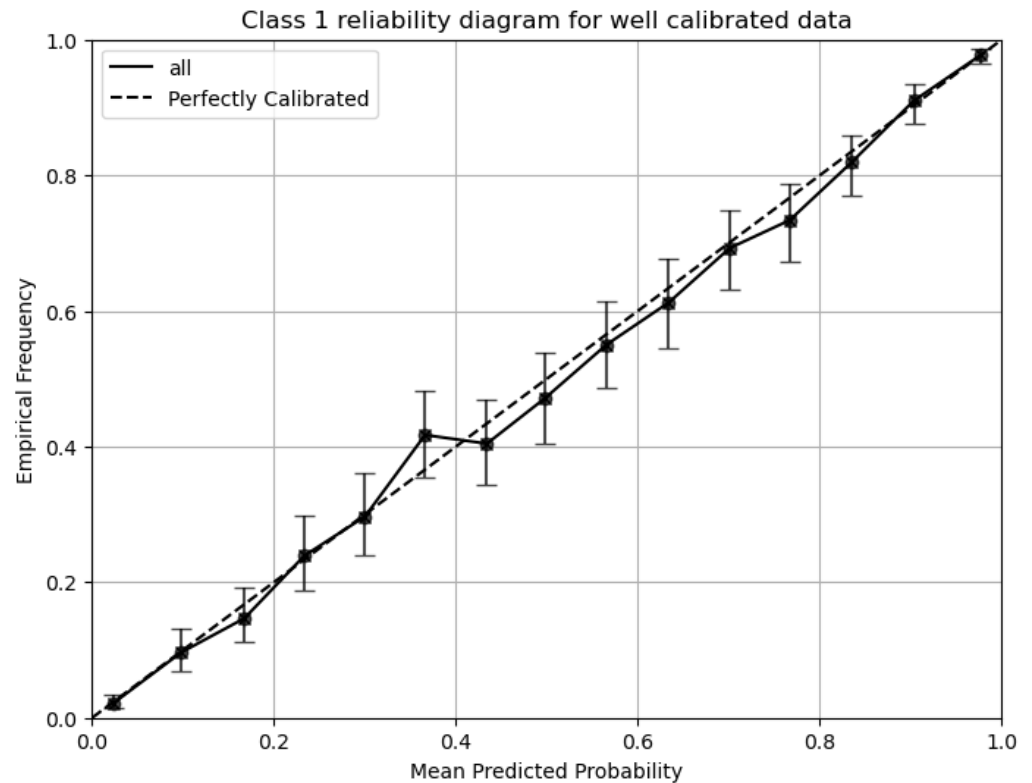


Figure 1: Reliability Diagram for well calibrated data

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
    )

```

60 Hosmer-Lemeshow statistic (HL)

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

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

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

```

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

79 Cox's calibration slope/intercept

80 Cox's calibration slope/intercept is a regression analysis method for assessing the calibration
 81 of a probabilistic model (COX, 1958). A new logistic regression model is fitted to the data,
 82 with the predicted odds ($\frac{p}{1-p}$) as the independent variable and the outcome as the dependent
 83 variable. The slope and intercept of the regression line are then used to assess the calibration
 84 of the model. A slope of 1 and intercept of 0 indicates perfect calibration. To test whether
 85 the model is calibrated, fix the slope to 1 and fit the intercept. If the intercept is significantly
 86 different from 0, the model is not calibrated. Then, fix the intercept to 0 and fit the slope.
 87 If the slope is significantly different from 1, the model is not calibrated. In calzone, Cox's
 88 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
)

```

89 «««< HEAD The slope and intercept values indicate the type of miscalibration. A slope >1
90 shows overconfidence at high probabilities and underconfidence at low probabilities (and vice
91 versa). A positive intercept indicates general overconfidence (and vice versa). However, even
92 with ideal slope and intercept values, the model may still be miscalibrated due to non-linear
93 effects that Cox's analysis cannot detect.

94 Integrated calibration index (ICI)

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

104 The values of the slope and intercept can represent miscalibration throughout the range of
105 probability outputs. The integrated calibration index (ICI) is very similar to Expected calibration
106 error (ECE). It also tries to measure the average deviation between predicted probability
107 and true probability. However, ICI does not use binning to estimate the true probability
108 of a group of samples with similar predicted probability. Instead, ICI uses curve smoothing
109 techniques to fit the regression curve and uses the regression result as the true probability
110 (Austin & Steyerberg, 2019). The ICI is then calculated using the following formula: »»»>
111 cdd3f8ac405788615653f94d7b4144af7e9acab0

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

112 where f is the fitting function and p is the predicted probability. The curve fitting is usually
113 done with loess regression. However, it is possible to use any curve fitting method to calculate
114 the ICI. In calzone, we provide Cox's ICI and loess ICI support while the user can also use any
115 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, 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
)
```

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

119 Spiegelhalter's Z-test

120 Spiegelhalter's Z-test is a test of calibration proposed by Spiegelhalter in 1986 (Spiegelhalter,
 121 1986). It uses the fact that the Brier score can be decomposed into:

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

122 And the TS of Z test is defined as:

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

123 and it is asymptotically distributed as a standard normal distribution. In calzone, it can be
 124 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
)
```

125 Metrics class

126 calzone also provides a class called CalibrationMetrics() to calculate all the metrics men-
 127 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'
)
```

Other features

Confidence intervals

In addition to point estimates of calibration performance, calzone also provides bootstrapping to calculate the confidence intervals of the metrics. 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_data_loader.labels,  
    wellcal_data_loader.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 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.

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.

158 Validation of methods

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

168 Command line interface

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

173 Acknowledgements

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

178 Conflicts of interest

179 The authors declare no conflicts of interest.

180 References

- 181 Austin, P. C., & Steyerberg, E. W. (2019). The integrated calibration index (ICI) and related
182 metrics for quantifying the calibration of logistic regression models. *Statistics in Medicine*,
183 38(21), 4051–4065. <https://doi.org/https://doi.org/10.1002/sim.8281>
- 184 Bröcker, J. (2009). Reliability, sufficiency, and the decomposition of proper scores. *Quarterly*
185 *Journal of the Royal Meteorological Society*, 135(643), 1512–1519. <https://doi.org/https://doi.org/10.1002/qj.456>
- 186 Bröcker, J., & Smith, L. A. (2007). Increasing the reliability of reliability diagrams. *Weather*
187 *and Forecasting*, 22(3), 651–661. <https://doi.org/10.1175/WAF993.1>
- 188 Calster, B. V., & Steyerberg, E. W. (2018). Calibration of prognostic risk scores. In
189 *Wiley StatsRef: Statistics reference online* (pp. 1–10). John Wiley & Sons, Ltd.
190 <https://doi.org/https://doi.org/10.1002/9781118445112.stat08078>
- 191 Chen, W., Sahiner, B., Samuelson, F., Pezeshk, A., & Petrick, N. (2018). Calibration of
192 medical diagnostic classifier scores to the probability of disease. *Statistical Methods in*
193 *Medical Research*, 27(5), 1394–1409. <https://doi.org/10.1177/0962280216661371>
- 194 Chung, Y., Char, I., Guo, H., Schneider, J., & Neiswanger, W. (2021). Uncertainty toolbox:
195 An open-source library for assessing, visualizing, and improving uncertainty quantification.
196 *arXiv Preprint arXiv:2109.10254*.
197

- 198 COX, D. R. (1958). Two further applications of a model for binary regression. *Biometrika*,
199 45(3-4), 562–565. <https://doi.org/10.1093/biomet/45.3-4.562>
- 200 Diamond, G. A. (1992). What price perfection? Calibration and discrimination of clinical
201 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)
202
- 203 Griffiths, D. A. (1973). Maximum likelihood estimation for the beta-binomial distribution
204 and an application to the household distribution of the total number of cases of a disease.
205 *Biometrics*, 29(4), 637–648. <http://www.jstor.org/stable/2529131>
- 206 Gu, W., & Pepe, M. S. (2010). Estimating the diagnostic likelihood ratio of a continuous
207 marker. *Biostatistics*, 12(1), 87–101. <https://doi.org/10.1093/biostatistics/kxq045>
- 208 Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural
209 networks. In D. Precup & Y. W. Teh (Eds.), *Proceedings of the 34th international*
210 *conference on machine learning* (Vol. 70, pp. 1321–1330). PMLR. <https://proceedings.mlr.press/v70/guo17a.html>
211
- 212 Horsch, K., Giger, M. L., & Metz, C. E. (2008). Prevalence scaling: Applications to an
213 intelligent workstation for the diagnosis of breast cancer. *Academic Radiology*, 15(11),
214 1446–1457. <https://doi.org/https://doi.org/10.1016/j.acra.2008.04.022>
- 215 Hosmer Jr, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). *Applied logistic regression*.
216 John Wiley & Sons.
- 217 Lele, S. R., Keim, J. L., & Solymos, P. (2024). *ResourceSelection: Resource selection (probabi-*
218 *lity) functions for use-availability data*. <https://github.com/psolymos/ResourceSelection>
- 219 Pakdaman Naeini, M., Cooper, G., & Hauskrecht, M. (2015). Obtaining well calibrated
220 probabilities using bayesian binning. *Proceedings of the AAAI Conference on Artificial*
221 *Intelligence*, 29(1). <https://doi.org/10.1609/aaai.v29i1.9602>
- 222 Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M.,
223 Prettenhofer, P., Weiss, R., Dubourg, V., & others. (2011). Scikit-learn: Machine learning
224 in python. *Journal of Machine Learning Research*, 12(Oct), 2825–2830.
- 225 Spiegelhalter, D. J. (1986). Probabilistic prediction in patient management and clinical trials.
226 *Statistics in Medicine*, 5(5), 421–433.
- 227 Taquet, V., Blot, V., Morzadec, T., Lacombe, L., & Brunel, N. (2022). MAPIE: An open-source
228 library for distribution-free uncertainty quantification. *arXiv Preprint arXiv:2207.12274*.
- 229 Tian, J., Liu, Y.-C., Glaser, N., Hsu, Y.-C., & Kira, Z. (2020). Posterior re-calibration
230 for imbalanced datasets. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan,
231 & H. Lin (Eds.), *Advances in neural information processing systems* (Vol. 33, pp.
232 8101–8113). Curran Associates, Inc. [https://proceedings.neurips.cc/paper_files/paper/](https://proceedings.neurips.cc/paper_files/paper/2020/file/5ca359ab1e9e3b9c478459944a2d9ca5-Paper.pdf)
233 [2020/file/5ca359ab1e9e3b9c478459944a2d9ca5-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2020/file/5ca359ab1e9e3b9c478459944a2d9ca5-Paper.pdf)
- 234 Wilson, E. B. (1927). Probable inference, the law of succession, and statistical inference.
235 *Journal of the American Statistical Association*, 22(158), 209–212.