




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Published: unpublished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 measuring calibration of probabilistic models for classification problems. It provides a set of functions and classes for calibration visualization and calibration metrics computation given a representative dataset with the model's predictions and the true labels. The metrics provided in calzone include the following: Expected Calibration Error (ECE), Maximum Calibration Error (MCE), Hosmer-Lemeshow statistic (HL), Integrated Calibration Index (ICI), Spiegelhalter's Z-statistics and Cox's calibration slope/intercept. Some metrics come with variations such as binning scheme and top-class or class-wise.

Statement of need

Classification is one of the most fundamental and important tasks in machine learning. The performance of classification models is often evaluated by a proper scoring rule, such as the cross-entropy or mean square error. Examination of the distinguishing power (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 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 defined 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. Most libraries for calibration are focusing on calibrating the model instead of measuring the level of calibration with various metrics. calzone is dedicated to calibration metrics computation and visualization.

Functionality

Reliability Diagram

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, we add the option to plot with error bars to show the confidence
41 interval of the empirical frequency in each bin. The error bars are calculated using Wilson's
42 score interval (Wilson, 1927). We provide an example simulated dataset in the example_data
43 folder using beta-binomial distribution (Griffiths, 1973). Users can generate simulated data
44 using the 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'
)
```

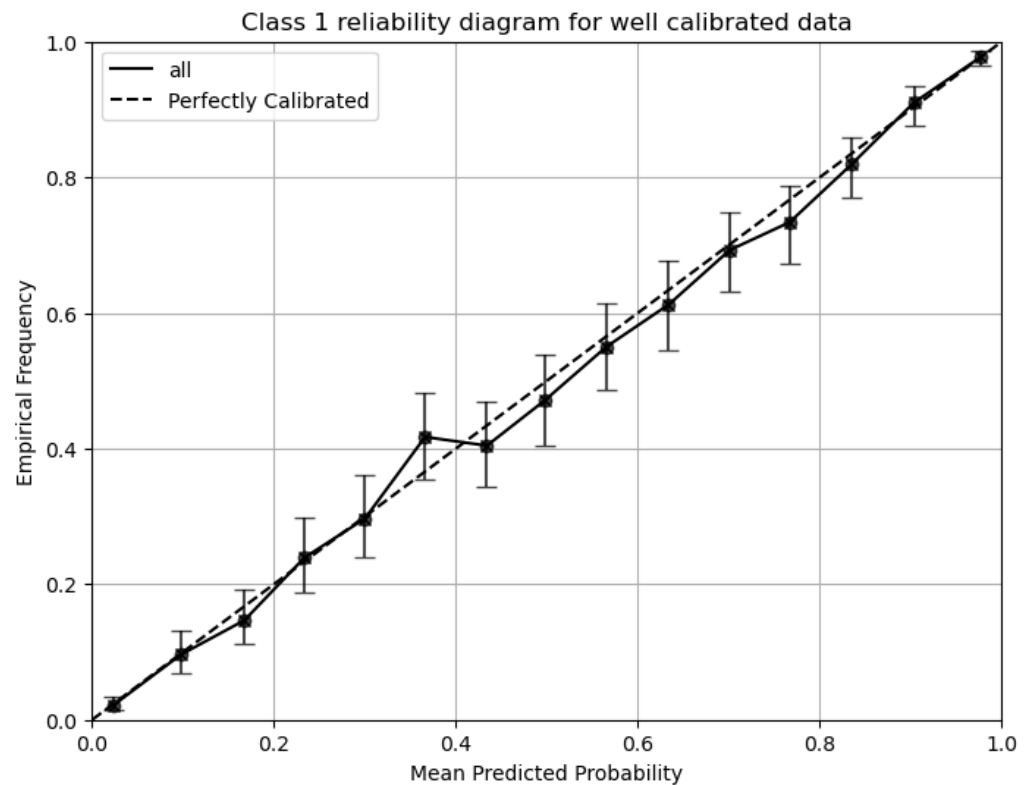


Figure 1: Reliability Diagram for well calibrated data

Calibration metrics

calzone provides functions to compute various calibration metrics. calzone also has a CalibrationMetrics() class which allows the user to compute the calibration metrics in a more convenient way. The following are the 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 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, the program will treat it as a 1-vs-rest classification problem. It can be computed in calzone as follows:

```
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 Hosmer-Lemeshow statistic (HL) is a statistical test for the calibration of a probabilistic model.
 59 It is a chi-square based test that compares the observed and expected number of events in
 60 each bin. The null hypothesis is that the model is well calibrated. HL-test first bins data into
 61 predicted probability bins (equal-width H or equal-count C) and the test statistic is calculated
 62 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 When performing the HL test on validation sets that are not used in training, the degree of
 68 freedom of the HL test changes from $M - 2$ to M . Intuitively, $\frac{(O_{1,m} - E_{1,m})^2}{E_{1,m}(1 - \frac{E_{1,m}}{N_m})}$ is the difference
 69 squared divided by the variance of a binomial distribution and follows a chi-square distribution
 70 with 1 degree of freedom. Hence, the sum of M chi-square distributions with 1 degree of
 71 freedom is a chi-square distribution with M degrees of freedom if the data has no effect on
 72 the model. In calzone, user can sepecify the degree of freedom of the HL test by setting the
 73 df parameter.

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 give you a sense of the form of miscalibration. A slope
85 greater than 1 indicates that the model is overconfident at high probabilities and underconfident
86 at low probabilities, and vice versa. An intercept greater than 0 indicates that the model is
87 overconfident in general, and vice versa. Notice that even if the slope is 1 and the intercept is
88 0, the model might not be calibrated, as Cox's calibration analysis fails to capture some types
89 of miscalibration, including quadratic effects or other non-linearities.

90 Integrated calibration index (ICI)

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

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

97 where f is the fitting function and p is the predicted probability. The curve fitting is usually
98 done with loess regression. However, it is possible to use any curve fitting method to calculate
99 the ICI. In calzone, we provide Cox's ICI and loess ICI support while the user can also use any
100 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_data_loader.labels,
    wellcal_data_loader.probs,
    class_to_calculate=1,
    span=0.5,
    delta=0.001,
    it=0
)
```

101 Notice that flexible curve fitting methods such as loess regression are very sensitive to the
102 choice of span and delta parameters. The user can visualize the fitting result to avoid overfitting
103 or underfitting.

104 **Spiegelhalter's Z-test**

105 Spiegelhalter's Z-test is a test of calibration proposed by Spiegelhalter in 1986 (Spiegelhalter,
106 1986). It uses the fact that the Brier score can be decomposed into:

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

107 And the TS of Z test is defined as:

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

108 and it is asymptotically distributed as a standard normal distribution. In calzone, it can be
109 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
)
```

110 **Metrics class**

111 calzone also provides a class called CalibrationMetrics() to calculate all the metrics men-
112 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'
)
```

113 **Other features**

114 **Bootstrapping**

115 calzone also provides bootstrapping to calculate the confidence intervals of the metrics. The
116 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
)
```

117 and it will return a structured numpy array.

118 Subgroup analysis

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

122 Prevalence adjustment

123 calzone also provides prevalence adjustment to account for prevalence changes between
124 training data and testing data. Since calibration is defined using posterior probability, a
125 mere shift in the prevalence of the testing data will result in miscalibration. It can be fixed
126 by searching for the optimal derived original prevalence such that the adjusted probability
127 minimizes a proper scoring rule such as cross-entropy loss. The formula of prevalence adjusted
128 probability is:

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

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

132 Multiclass extension

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

142 Command line interface

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

147 Acknowledgements

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

152 Conflicts of interest

153 The authors declare no conflicts of interest.

154 References

155 Austin, P. C., & Steyerberg, E. W. (2019). The integrated calibration index (ICI) and related
156 metrics for quantifying the calibration of logistic regression models. *Statistics in Medicine*,

- 157 38(21), 4051–4065. <https://doi.org/https://doi.org/10.1002/sim.8281>
- 158 Bröcker, J. (2009). Reliability, sufficiency, and the decomposition of proper scores. *Quarterly*
159 *Journal of the Royal Meteorological Society*, 135(643), 1512–1519. <https://doi.org/https://doi.org/10.1002/qj.456>
- 160
- 161 Bröcker, J., & Smith, L. A. (2007). Increasing the reliability of reliability diagrams. *Weather*
162 *and Forecasting*, 22(3), 651–661. <https://doi.org/10.1175/WAF993.1>
- 163 Calster, B. V., & Steyerberg, E. W. (2018). Calibration of prognostic risk scores. In
164 *Wiley StatsRef: Statistics reference online* (pp. 1–10). John Wiley & Sons, Ltd.
165 <https://doi.org/https://doi.org/10.1002/9781118445112.stat08078>
- 166 Chen, W., Sahiner, B., Samuelson, F., Pezeshk, A., & Petrick, N. (2018). Calibration of
167 medical diagnostic classifier scores to the probability of disease. *Statistical Methods in*
168 *Medical Research*, 27(5), 1394–1409. <https://doi.org/10.1177/0962280216661371>
- 169 COX, D. R. (1958). Two further applications of a model for binary regression. *Biometrika*,
170 45(3-4), 562–565. <https://doi.org/10.1093/biomet/45.3-4.562>
- 171 Griffiths, D. A. (1973). Maximum likelihood estimation for the beta-binomial distribution
172 and an application to the household distribution of the total number of cases of a disease.
173 *Biometrics*, 29(4), 637–648. <http://www.jstor.org/stable/2529131>
- 174 Gu, W., & Pepe, M. S. (2010). Estimating the diagnostic likelihood ratio of a continuous
175 marker. *Biostatistics*, 12(1), 87–101. <https://doi.org/10.1093/biostatistics/kxq045>
- 176 Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural
177 networks. In D. Precup & Y. W. Teh (Eds.), *Proceedings of the 34th international*
178 *conference on machine learning* (Vol. 70, pp. 1321–1330). PMLR. <https://proceedings.mlr.press/v70/guo17a.html>
- 179
- 180 Horsch, K., Giger, M. L., & Metz, C. E. (2008). Prevalence scaling: Applications to an
181 intelligent workstation for the diagnosis of breast cancer. *Academic Radiology*, 15(11),
182 1446–1457. <https://doi.org/https://doi.org/10.1016/j.acra.2008.04.022>
- 183 Pakdaman Naeini, M., Cooper, G., & Hauskrecht, M. (2015). Obtaining well calibrated
184 probabilities using bayesian binning. *Proceedings of the AAAI Conference on Artificial*
185 *Intelligence*, 29(1). <https://doi.org/10.1609/aaai.v29i1.9602>
- 186 Spiegelhalter, D. J. (1986). Probabilistic prediction in patient management and clinical trials.
187 *Statistics in Medicine*, 5(5), 421–433.
- 188 Tian, J., Liu, Y.-C., Glaser, N., Hsu, Y.-C., & Kira, Z. (2020). Posterior re-calibration
189 for imbalanced datasets. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan,
190 & H. Lin (Eds.), *Advances in neural information processing systems* (Vol. 33, pp.
191 8101–8113). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2020/file/5ca359ab1e9e3b9c478459944a2d9ca5-Paper.pdf
- 192
- 193 Wilson, E. B. (1927). Probable inference, the law of succession, and statistical inference.
194 *Journal of the American Statistical Association*, 22(158), 209–212.