The Journal of Open Source Software

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

Kwok Lung Fan [1], **Gene Pennello** [1], **Qi Liu** [1], **Nicholas Petrick** [1], **Ravi K. Samala** [1], **Frank W. Samuelson** [1], **Yee Lam Elim Thompson** [1], and **Qian Cao** [1¶]

1 U.S. Food and Drug Administration ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

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 computing of calibration metrics given a representative dataset with the model's predictions and the 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 may output predicted probabilities that an input observation belongs to a particular class, and those probabilities 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 model performance. These metrics may be sufficient if the output of the model is not meant to be a calibrated probability.

Diamond (1992) showed that the resolution (i.e., high performance) of a model does not indicate the reliability/calibration (i.e., how well predicted probabilities match true probabilities) of the model. Bröcker (2009) later showed that any proper scoring rule can be decomposed into the resolution and reliability. Thus, even if the model has high resolution, it may not be a reliable model. In many high-risk machine learning applications, such as medical diagnosis and prognosis, reliability greatly aids in the interpretability of the model when deciding among multiple treatment options.

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) and is also referred as the reliability of the model.

In the calzone package, we provide a set of functions and classes for visualizing calibration and evaluating calibration metrics given a representative dataset from the intended population. Existing libraries such as `scikit-learn` lacks calibration metrics that are widely used in the statistical literature. Other libraries such as `uncertainty-toolbox` are focused on implementing calibration methods and not calibration assessment. (Chung et al., 2021).

Software description

Input data

To evaluate the calibration of a model, users need a representative dataset from the intended population. The dataset should contain the true class labels and the model's predicted probabilities. In `calzone`, the dataset can be loaded from a CSV file using the `data_loader` function. The description of the input CSV file format can be found in the `calzone` documentation. Alternatively, users can pass the true class labels and the model's predicted probabilities as NumPy arrays to the `calzone` functions.

Reliability Diagram

The reliability diagram (also referred to as the calibration plot) is a graphical representation of the calibration of a classification model (Bröcker & Smith, 2007; Murphy & Winkler, 1977). It groups the predicted probabilities into bins and plots the mean predicted probability against the empirical frequency in each bin. The reliability diagram can be used to qualitatively assess the calibration of the model and to identify any systematic errors in the predictions. In addition, `calzone` gives the option to also plot the confidence interval of the empirical frequency in each bin. The confidence intervals are calculated using the Wilson's score interval (Wilson, 1927).

We provide example data in the `example_data` folder which are simulated using a beta-binomial distribution (Griffiths, 1973). More description can be found in the documentation. In the example code below, we will instead use the scikit-learn function to simulate a random 2-class dataset of correlated normally distributed data and fit a simple logistic model to illustrate. Figure 1 shows an example of the reliability diagram for class 1 with 15 equal-width bins for a well-calibrated dataset, where the x-axis is the mean predicted probability and the y-axis is the empirical frequency.

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

# Generate a random binary classification dataset
import sklearn.linear_model
import sklearn.datasets

features, labels = sklearn.datasets.make_classification(
    n_samples=5000,
    n_features=5,
    n_informative=5,
    n_redundant=0,
    n_classes=2,
    n_clusters_per_class=1,
    class_sep=0.2,
    random_state=1
)
model = sklearn.linear_model.LogisticRegression()
model.fit(features, labels)
probs = model.predict_proba(features)

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

```python
plot_reliability_diagram(
    reliability,
    confidence,
    bin_counts,
    error_bar=True,
    title='Reliability diagram'
)
```

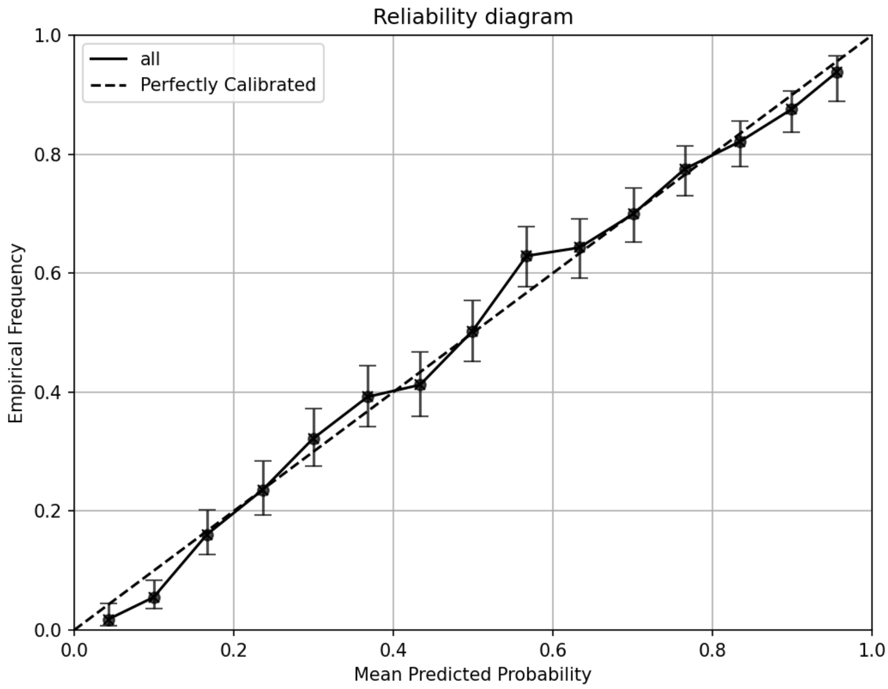

Figure 1: Reliability Diagram for class 1 with simulated data.

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) and Maximum Calibration Error (MCE) (Guo et al., 2017; Pakdaman Naeini et al., 2015) aim to measure the average and maximum absolute deviation between the 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. Top-class mean only the calibration of the class with the highest probability is evaluted. 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:

```python
from calzone.metrics import calculate_ece_mce
```

```python
reliability, confidence, bin_edges, bin_counts = reliability_diagram(
    labels,
    probs,
    num_bins=10,
    class_to_plot=1,
    is_equal_freq=False
)
### Both ECE and MCE are calculated at the same time
ece_h_classone, mce_h_classone = calculate_ece_mce(
    reliability,
    confidence,
    bin_counts=bin_counts
)
```

Hosmer-Lemeshow statistic (HL)

The Hosmer-Lemeshow (HL) statistical test (Hosmer & Lemesbow, 1980) is for evaluating the calibration of a probabilistic model. It is a chi-square-based test that compares the observed and expected number of events in each bin. The null hypothesis is that the model is well calibrated. HL-test first bins data into predicted probability bins (equal-width H or equal-count C) and the test statistic is calculated as:

$$\text{HL} = \sum_{m=1}^{M} \frac{(O_{1,m} - E_{1,m})^2}{E_{1,m}\left(1 - \frac{E_{1,m}}{N_m}\right)} \sim \chi^2_{M-2}$$

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

```python
from calzone.metrics import hosmer_lemeshow_test

HL_H_ts, HL_H_p, df = hosmer_lemeshow_test(
    reliability,
    confidence,
    bin_count = bin_counts,
    df = len(bin_counts) - 2,
)
```

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

Cox's calibration slope/intercept

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

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

Integrated calibration index (ICI)

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

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

where f is the fitting function and p is the predicted probability. The curve fitting is usually done with Locally Weighted Scatterplot Smoothing (LOWESS). However, it is possible to use any curve fitting method to calculate the ICI. One possible altenatively is to use the Cox calibration result and calculate the average difference between the predicted probability and the estimated true probability from the curve. In calzone, we provide the Cox ICI and LOWESS ICI support while the user can also use any 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,
    probs,
```

```python
    class_to_calculate=1
)

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

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 test statistic (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:

```python
from calzone.metrics import spiegelhalter_z_test

z, p_value = spiegelhalter_z_test(
    labels,
    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 a list of metrics or all the metrics within a single function call. The function will return a dictionary containing the metrics name and their values. The metrics can be specified as a list of strings. The string 'all' can be used to calculate all the metrics.

```python
from calzone.metrics import CalibrationMetrics

metrics = CalibrationMetrics(class_to_calculate=1)

metrics.calculate_metrics(
    labels,
    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 only exception is the confidence intervals from the reliability diagram which calculates Wilson's score intervals. 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(
    labels,
    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. A detailed description of the input format can be found in the documentation.

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 also specify η' and adjust the probability output directly if the training set prevalence is available.

Multiclass extension

`calzone` provides a multiclass extension 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 1 probability is defined as the probability of the class with the highest probability and the class 0 probability is defined as 1 minus the probability of the class with the highest probability. The labels are transformed into whether the predicted class equals the true highest probability class, 0 if not and 1 if yes. Notice that the interpretation of some metrics may change in the top-class transformation because the probability of class 1 after transformation is not tied to a specific class in the original dataset.

Verification of methods

We compared the results calculated by `calzone` with external packages for some metrics to ensure the correctness of the implementation. For the reliability diagram verification, we compared the result with the `sklearn.calibration.calibration_curve()` function in `scikit-learn` (Pedregosa et al., 2011) with the reliability diagram produced by calzone. For the top-class ECE and SpiegelHalter's Z scores, we compared the result with the `MAPIE` package (Taquet et al., 2022). For the Hosmer-Lemeshow statistic, we compared the result with the `ResourceSelection` package in R language (Lele et al., 2024). The difference in result for all functions tested were within 0.1%, indicating calzone output values are very consistent with the values produced by other packages. We include the verification codes and comparison 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 this. For help on running this functionality, the user can run `python cal_metrics.py -h`.

Acknowledgements

The mention of commercial products, their sources, or their use in connection with material reported herein is not to be construed as either an actual or implied endorsement of such products by the Department of Health and Human Services. This is a contribution of the U.S. Food and Drug Administration and is not subject to copyright.

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.

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

Chen, W., Sahiner, B., Samuelson, F., Pezeshk, A., & Petrick, N. (2018). Calibration of medical diagnostic classifier scores to the probability of disease. *Statistical Methods in Medical Research*, *27*(5), 1394–1409. https://doi.org/10.1177/0962280216661371

Chung, Y., Char, I., Guo, H., Schneider, J., & Neiswanger, W. (2021). Uncertainty toolbox: An open-source library for assessing, visualizing, and improving uncertainty quantification. *arXiv Preprint arXiv:2109.10254*.

Cox, D. R. (1958). Two further applications of a model for binary regression. *Biometrika*, *45*(3-4), 562–565. https://doi.org/10.1093/biomet/45.3-4.562

Diamond, G. A. (1992). What price perfection? Calibration and discrimination of clinical prediction models. *Journal of Clinical Epidemiology*, *45*(1), 85–89. https://doi.org/10.1016/0895-4356(92)90192-P

Gneiting, T., & Raftery, A. E. (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, *102*(477), 359–378.

Griffiths, D. A. (1973). Maximum likelihood estimation for the beta-binomial distribution and an application to the household distribution of the total number of cases of a disease. *Biometrics*, *29*(4), 637–648. http://www.jstor.org/stable/2529131

Gu, W., & Pepe, M. S. (2010). Estimating the diagnostic likelihood ratio of a continuous marker. *Biostatistics*, *12*(1), 87–101. https://doi.org/10.1093/biostatistics/kxq045

Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural networks. In D. Precup & Y. W. Teh (Eds.), *Proceedings of the 34th international conference on machine learning* (Vol. 70, pp. 1321–1330). PMLR. https://proceedings.mlr.press/v70/guo17a.html

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/10.1016/j.acra.2008.04.022

Hosmer, D. W., & Lemesbow, S. (1980). Goodness of fit tests for the multiple logistic regression model. *Communications in Statistics - Theory and Methods*, *9*(10), 1043–1069. https://doi.org/10.1080/03610928008827941

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/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.