




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Published: unpublished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 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. However, the reliability or calibration performance of the model is often overlooked.

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).

41 Software description

42 Input data

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

49 Reliability Diagram

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

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

```
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
)
```

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

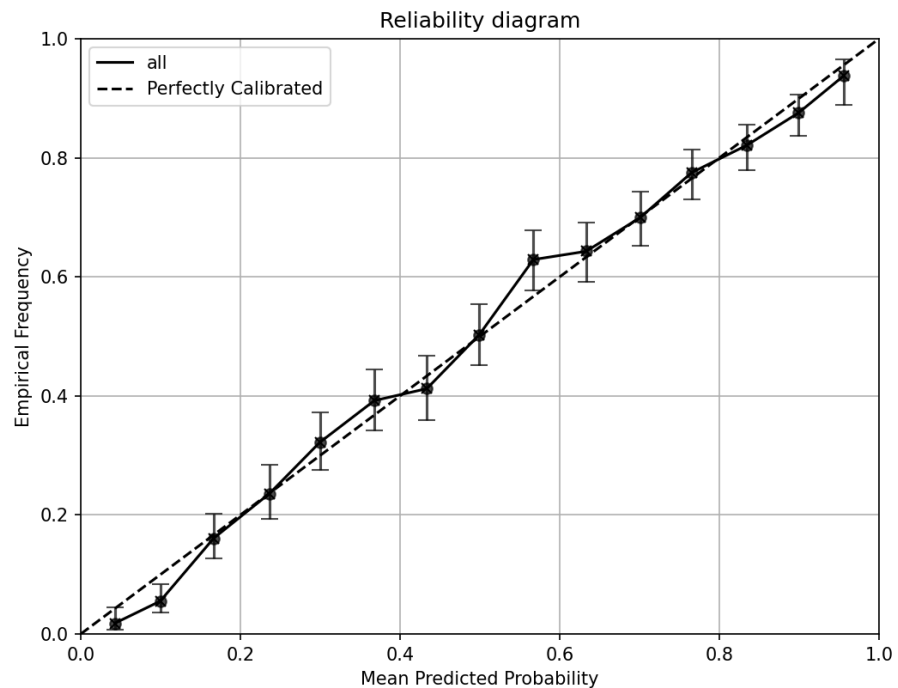


Figure 1: Reliability Diagram for class 1 with simulated data.

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 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 predicted class will be evaluated. 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(
```

```

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
)

```

76 Hosmer-Lemeshow statistic (HL)

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

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

82 where $E_{1,m}$ is the expected number of class-of-interest events in the m^{th} bin, $O_{1,m}$ is the
 83 observed number of class-of-interest events in the m^{th} bin, N_m is the total number of
 84 observations in the m^{th} bin, and M is the number of bins. In calzone, the HL-test can be
 85 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,
    df = len(bin_counts) - 2,
)

```

86 When performing the HL test on validation sets that are not used in training, the degree
 87 of freedom of the HL test changes from $M - 2$ to M (Hosmer Jr et al., 2013). Intuitively,
 88 $\frac{(O_{1,m} - E_{1,m})^2}{E_{1,m} \left(1 - \frac{E_{1,m}}{N_m}\right)}$ is the difference squared divided by the variance of a binomial distribution and
 89 follows a chi-square distribution with 1 degree of freedom. Hence, the sum of M chi-square
 90 distributions with 1 degree of freedom is a chi-square distribution with M degrees of freedom if
 91 the data has no effect on the model. The increase in degree of freedom for validation samples
 92 has often been overlooked but it is crucial for the test to maintain the correct type 1 error
 93 rate. In calzone, the default degree of freedoms is $M - 2$ and users should specify the degree
 94 of freedom of the HL test by setting the `df` parameter.

95 Cox's calibration slope/intercept

96 Cox's calibration slope/intercept is a regression analysis method for assessing the calibration
 97 of a probabilistic model (COX, 1958), which doesn't require binning. A logistic regression
 98 model is fit to the data, with the predicted odds $\left(\frac{p}{1-p}\right)$ as the independent variable and the
 99 outcome as the dependent variable. The slope and intercept of the regression line are then
 100 used to assess the calibration of the model. A slope of 1 and intercept of 0 indicates perfect

101 calibration. To test whether the model is calibrated, fix the slope to 1 and fit the intercept.
102 If the intercept is significantly different from 0, the model is not calibrated. Then, fix the
103 intercept to 0 and fit the slope. If the slope is significantly different from 1, the model is not
104 calibrated. Alternatively, the slope and intercept can be fitted and tested simultaneously using
105 a bivariate distribution (McCullagh & Nelder, 1989). This feature is not provided in calzone
106 but user can extract the covariance matrix by printing the result and perform the test manually.
107 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
)
```

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

114 Integrated calibration index (ICI)

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

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

121 where f is the fitting function and p is the predicted probability. The curve fitting is usually
122 done with Locally Weighted Scatterplot Smoothing (LOWESS). However, it is possible to use
123 any curve fitting method to calculate the ICI. One possible alternative is to use the Cox's
124 calibration result and calculate the average difference between the predicted probability and
125 the estimated true probability from the curve. In calzone, we provide Cox's ICI and loess
126 ICI support while the user can also use any curve fitting method to calculate the ICI using
127 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,
    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
)
```

128 Notice that flexible curve fitting methods such as LOWESS regression are very sensitive to
129 the choice of span and delta parameters. The user can visualize the fitting result to avoid
130 overfitting or underfitting.

131 Spiegelhalter's Z-test

132 Spiegelhalter's Z-test is a test of calibration proposed by Spiegelhalter in 1986 (Spiegelhalter,
133 1986). It uses the fact that the Brier score can be decomposed into:

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

134 And the test statistic (TS) of Z test is defined as:

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

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

```
from calzone.metrics import spiegelhalter_z_test

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

137 Metrics class

138 calzone also provides a class called CalibrationMetrics() to calculate all the metrics men-
139 tioned above. The user can also use this class to calculate a list of metrics or all the metrics
140 within a single function call. The function will return a dictionary containing the metrics name
141 and their values. The metrics can be specified as a list of string. The string 'all' can be used
142 to calculate all the metrics.

```
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. 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 also provides a 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.

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). 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 are within 0.1%, indicating that implementations are essentially the same. For other metrics such as ICI, no external package is available, so we compared the result with ECE as they both measure the average absolute difference. We obtained reasonably similar results. 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.

This article reflects the views of the author and should not be construed to represent FDA's views or policies

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/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/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>

- 217 Chung, Y., Char, I., Guo, H., Schneider, J., & Neiswanger, W. (2021). Uncertainty toolbox:
218 An open-source library for assessing, visualizing, and improving uncertainty quantification.
219 *arXiv Preprint arXiv:2109.10254*.
- 220 COX, D. R. (1958). Two further applications of a model for binary regression. *Biometrika*,
221 45(3-4), 562–565. <https://doi.org/10.1093/biomet/45.3-4.562>
- 222 Diamond, G. A. (1992). What price perfection? Calibration and discrimination of clinical
223 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)
- 224
- 225 Gneiting, T., & Raftery, A. E. (2007). Strictly proper scoring rules, prediction, and estimation.
226 *Journal of the American Statistical Association*, 102(477), 359–378.
- 227 Griffiths, D. A. (1973). Maximum likelihood estimation for the beta-binomial distribution
228 and an application to the household distribution of the total number of cases of a disease.
229 *Biometrics*, 29(4), 637–648. <http://www.jstor.org/stable/2529131>
- 230 Gu, W., & Pepe, M. S. (2010). Estimating the diagnostic likelihood ratio of a continuous
231 marker. *Biostatistics*, 12(1), 87–101. <https://doi.org/10.1093/biostatistics/kxq045>
- 232 Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural
233 networks. In D. Precup & Y. W. Teh (Eds.), *Proceedings of the 34th international*
234 *conference on machine learning* (Vol. 70, pp. 1321–1330). PMLR. <https://proceedings.mlr.press/v70/guo17a.html>
- 235
- 236 Horsch, K., Giger, M. L., & Metz, C. E. (2008). Prevalence scaling: Applications to an
237 intelligent workstation for the diagnosis of breast cancer. *Academic Radiology*, 15(11),
238 1446–1457. <https://doi.org/https://doi.org/10.1016/j.acra.2008.04.022>
- 239 Hosmer, D. W., & Lemeshow, S. (1980). Goodness of fit tests for the multiple logistic
240 regression model. *Communications in Statistics - Theory and Methods*, 9(10), 1043–1069.
241 <https://doi.org/10.1080/03610928008827941>
- 242 Hosmer Jr, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). *Applied logistic regression*.
243 John Wiley & Sons.
- 244 Lele, S. R., Keim, J. L., & Solymos, P. (2024). *ResourceSelection: Resource selection (prob-*
245 *ability) functions for use-availability data*. <https://github.com/psolymos/ResourceSelection>
- 246 McCullagh, P., & Nelder, J. A. (1989). *Generalized linear models*. Chapman & Hall / CRC.
- 247 Murphy, A. H., & Winkler, R. L. (1977). Reliability of subjective probability forecasts of
248 precipitation and temperature. *Journal of the Royal Statistical Society. Series C (Applied*
249 *Statistics)*, 26(1), 41–47. <http://www.jstor.org/stable/2346866>
- 250 Pakdaman Naeini, M., Cooper, G., & Hauskrecht, M. (2015). Obtaining well calibrated
251 probabilities using bayesian binning. *Proceedings of the AAAI Conference on Artificial*
252 *Intelligence*, 29(1). <https://doi.org/10.1609/aaai.v29i1.9602>
- 253 Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M.,
254 Prettenhofer, P., Weiss, R., Dubourg, V., & others. (2011). Scikit-learn: Machine learning
255 in python. *Journal of Machine Learning Research*, 12(Oct), 2825–2830.
- 256 Spiegelhalter, D. J. (1986). Probabilistic prediction in patient management and clinical trials.
257 *Statistics in Medicine*, 5(5), 421–433.
- 258 Taquet, V., Blot, V., Morzadec, T., Lacombe, L., & Brunel, N. (2022). MAPIE: An open-source
259 library for distribution-free uncertainty quantification. *arXiv Preprint arXiv:2207.12274*.
- 260 Tian, J., Liu, Y.-C., Glaser, N., Hsu, Y.-C., & Kira, Z. (2020). Posterior re-calibration
261 for imbalanced datasets. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan,
262 & H. Lin (Eds.), *Advances in neural information processing systems* (Vol. 33, pp.

- 263 8101–8113). Curran Associates, Inc. [https://proceedings.neurips.cc/paper_files/paper/](https://proceedings.neurips.cc/paper_files/paper/2020/file/5ca359ab1e9e3b9c478459944a2d9ca5-Paper.pdf)
264 [2020/file/5ca359ab1e9e3b9c478459944a2d9ca5-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2020/file/5ca359ab1e9e3b9c478459944a2d9ca5-Paper.pdf)
- 265 Van Calster, B., & Steyerberg, E. W. (2018). Calibration of prognostic risk scores. In
266 *Wiley StatsRef: Statistics reference online* (pp. 1–10). John Wiley & Sons, Ltd.
267 <https://doi.org/https://doi.org/10.1002/9781118445112.stat08078>
- 268 Wilson, E. B. (1927). Probable inference, the law of succession, and statistical inference.
269 *Journal of the American Statistical Association*, 22(158), 209–212.

DRAFT