Predictive Performance Assessment

Nikolas Krstic
Applied Statistics and Data Science Group (ASDa)
Department of Statistics, UBC

July 29&30, 2024

Outline

- Performance measures
- Validation using hold-out set
- Cross-validation
- Validation with hyperparameter selection or model selection
- Boostrap validation

Predictive Performance Assessment Overview

When conducting any supervised learning or predictive modelling, it is absolutely important to appropriately estimate the predictive performance of the method.

Want to avoid misreporting or overexaggerating the predictive ability of the model, such as by reporting only the performance on the training set.

Multiple different types of performance measures can be used depending on the type of problem.

Multiple different methods to assess the predictive performance itself.

Predictive Performance Measures - Problem Type

There are numerous predictive performance measures that are available, but one key characteristic that helps narrow down the possible measures to use is the type of problem:

- Classification (Categorical/Binary Outcome)
- Regression (Continuous Outcome)

Additionally, there is no "optimal" measure of performance. The choice of performance measure will heavily depend on the problem at hand (i.e. what aspects of the prediction problem are considered important).

Somewhat connected to the ideas of "loss" and "risk" (you may have heard of this in machine learning). The "loss" is incurred by making bad predictions or decisions.

Examples of Different Performance Objectives

- Diagnosis Test/Model - False "negatives" (disease present, test doesn't detect) are much more harmful than false "positives" (disease absent, test indicates presence).
- Predict Bus Arrival Times - Would rather underestimate the time than overestimate the time it takes since the previous bus.
- Air Quality Concentrations - Want to avoid being "too" incorrect (i.e. want to penalize large differences between the prediction and truth even more than "usual").

Binary Outcome - Performance Measures

With a binary outcome, we predict whether an observation belongs to one of two possible categories, based on the available data (i.e. features or predictors).

Often can assign one category to be the "positive" class and another to be the "negative" class.

The category that indicates "presence"/"success" or is the category of greatest interest is the "positive" class (sometimes arbitrary though).

Nearly all performance measures for binary outcomes are built around proportions or counts of correct/incorrect classifications for positive/negative observations.

Predicting Binary Outcome - Four possible occurrences:

Below is what is known as a **confusion matrix**:

		Prediction	
		Positive	Negative
Truth	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Two of the most basic measures are:

- Sensitivity/True Positive Rate (TPR)/Recall:

$$\frac{TP}{TP+FN}$$

- Specificity/True Negative Rate (TNR):

$$\frac{TN}{TN+FP}$$

Sensitivity and Specificity

Both sensitivity and specificity are on a scale of 0 to 1, with 1 being the optimal value for both.

Perfect sensitivity means we can always correctly identify any positive observation using our model. Same for perfect specificity regarding negative observations.

However, there is always an inherent trade-off between the two (greater sensitivity leads to less specificity, and vice versa).

Very rare to achieve both very high sensitivity and specificity, so a balance must be struck between the two based on which has greater importance and by how much.

More Binary Outcome Performance Measures

Nearly every predictive performance measure for binary outcomes is computed from one or more of the counts (TP,FP,TN,FN) in the confusion matrix.

Here is a list of **some** of the other performance measures for binary outcomes:

- Accuracy - $\frac{TP+TN}{TP+FP+TN+FN}$
- Positive Predictive Value (PPV)/Precision - $\frac{TP}{TP+FP}$
- Negative Predictive Value (NPV) - $\frac{TN}{TN+FN}$
- Matthew's Correlation Coefficient (MCC):

$$\frac{\sqrt{TPR \times TNR \times PPV \times NPV} - \sqrt{(1-TPR) \times (1-TNR) \times (1-PPV) \times (1-NPV)}}{}$$

Class Probability Thresholds

For some models/tests sensitivity and specificity will be fixed, but some models assign predictions using "class probabilities" (e.g., logistic regression, random forests, etc.).

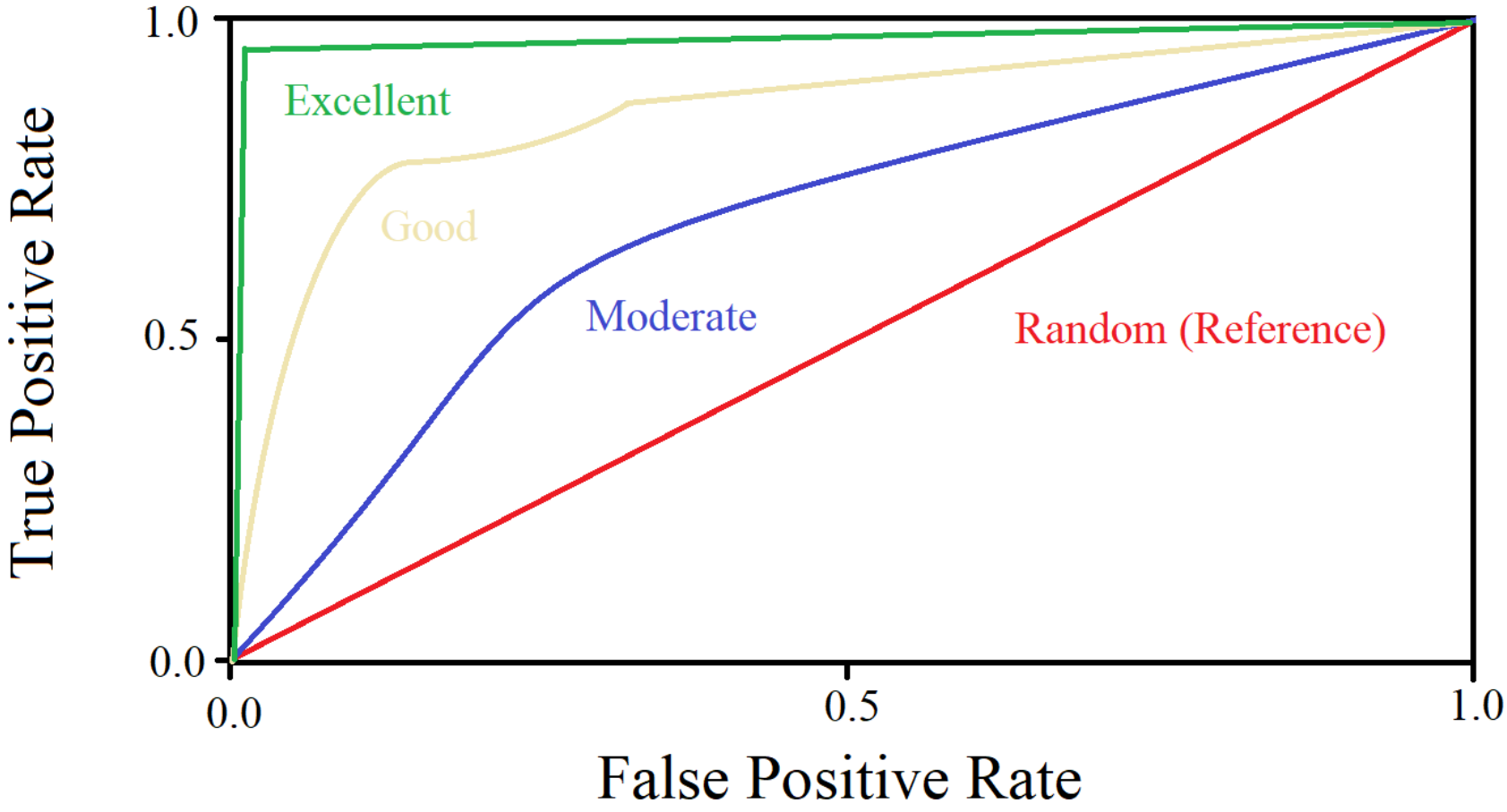
Therefore, we can manually specify the class probability threshold. Any observation whose class probability is predicted to be above the threshold is predicted to be "positive", otherwise "negative".

In turn, the problem is that computing any of the previous performance measures requires selection of the threshold. Couple of options:

- The threshold is selected in advance (usually results in poor final performance, if choosing something arbitrary like 0.5)
- The threshold becomes another hyperparameter for the model (doable, but can make performance assessment slower)

ROC Curves

Third way to address thresholding is to construct a Receiver-Operating Characteristic (ROC) Curve:



ROC Curves Explained

For each threshold that **uniquely** splits the observations of the training/test dataset, corresponds to a point on the ROC curve (i.e. can compute a unique TPR and FPR).

Start with maximum threshold 1 such that all observations are classified as negative (TPR=0 and FPR=0), then slowly decrease threshold to determine unique splits as some observations begin to be classified as positive.

Note that curve is always monotonic increasing, since either TPR or FPR increase:

- Positive Obs. Passes Threshold $\rightarrow$ TPR increase $\rightarrow$ Go Up
- Negative Obs. Passes Threshold $\rightarrow$ FPR increase $\rightarrow$ Go Right

ROC Curve Interpretation

Ideal binary classifier aims to have its ROC curve get as close as possible to the top left corner (such that TPR=1 and FPR=0 for one of the particular thresholds).

Area under the Curve (AUC) usually computed for ROC curves to gauge overall quality of predictive performance:

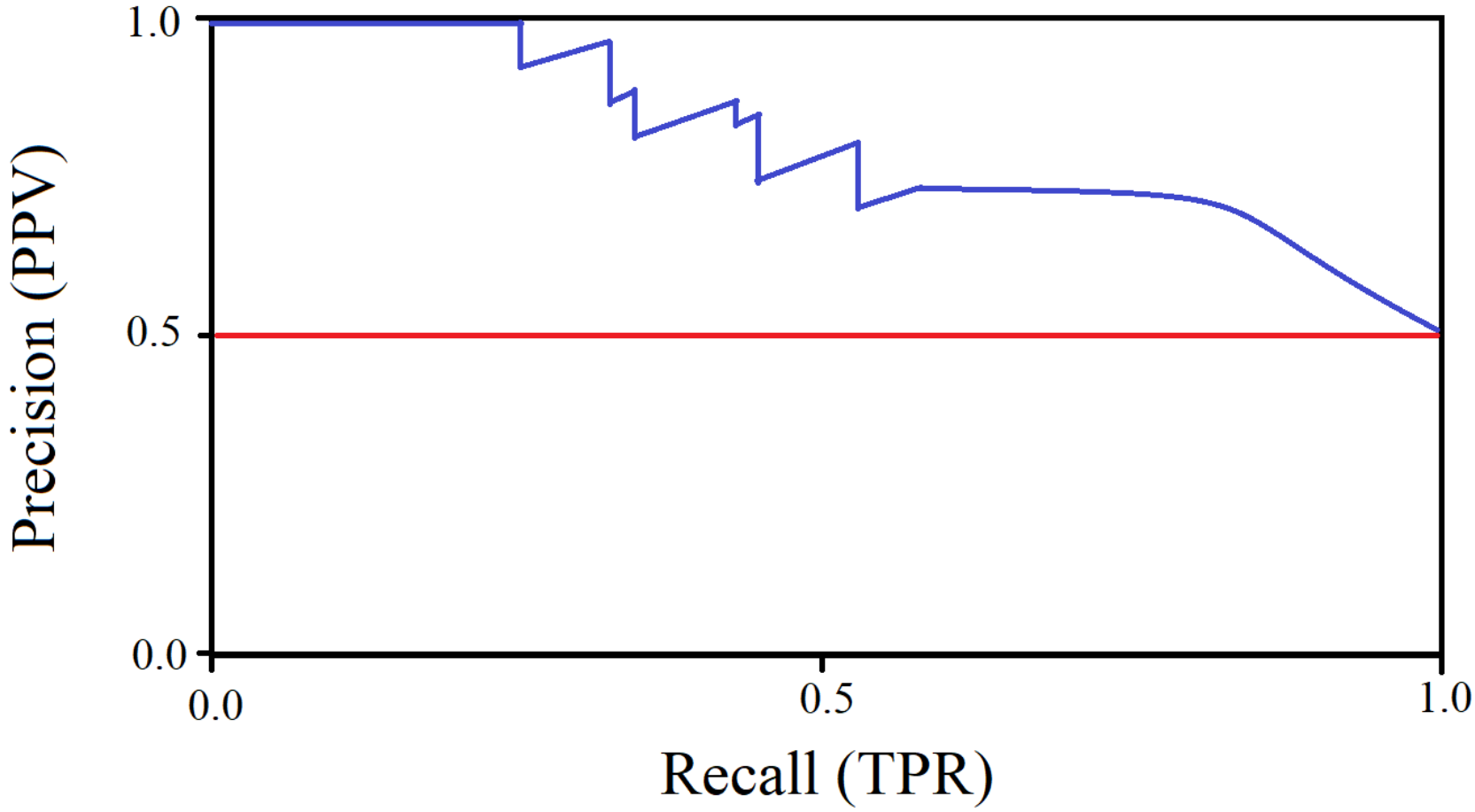
- ROC-AUC $= 0.5 \rightarrow$ Classifier is no better than **random**
- ROC-AUC $= 0.6$ - $0.7 \rightarrow$ **Moderate** Performance
- ROC-AUC $= 0.8$ - $0.9 \rightarrow$ **Good** to **Excellent** Performance
- ROC-AUC $= 1 \rightarrow$ **Perfect** Performance

Very good way to assess predictive performance of a model without having to select or tune a specific threshold.

ASIDE: Can get AUC-ROC below 0.5 too, which is often indicative of something extremely peculiar going on with the modelling, but can also happen by chance.

Precision-Recall Curves

Similar type of curve as ROC curve, but uses Recall (TPR) and Precision (PPV) instead. Useful for problems where "positives" are rare.



Precision-Recall Curves Disadvantages

Precision-Recall (P-R) curves have a few disadvantages:

- Classification of first observation that passes threshold is critical, because it dictates whether curve starts at 1 or 0.
- AUC of P-R curves can be difficult to interpret, since the "reference" is now based on the class ratio (i.e. what is the proportion of positives?).
- Model comparisons using P-R curves can be difficult because of the first disadvantage (AUC of P-R curves can be very different).
- Not really as useful when positives and negatives are balanced or when negatives are rarer (in fact, don't recommend to use at all in this latter case).

General Remarks on Performance Curves

Generally recommend only using P-R curves **in addition** to ROC curves, to get additional context on performance.

When positives are rare, they can be incredibly useful because ROC curves may suggest really good performance, but the P-R curve may indicate "too many" negatives are being falsely classified as positive based on PPV.

Can generalize ideas behind ROC and P-R curves to make similar curves for multiple other performance measures of interest:

- Make a grid of threshold values (e.g., 0.01 to 1, taking steps of 0.01).
- Compute the performance measure for each threshold value.
- Plot curve
- Repeat previous steps for each measure of interest.

Class Imbalance Problem

Need to be careful about potential class imbalances in the dataset(s), as this will affect interpretation (and potentially computation) of predictive performance measures.

If 99% of observations are negative and the remaining 1% are positive, then a model that **always** predicts an observation to be negative will achieve 99% accuracy.

This is why **accuracy** is often an atrocious measure for imbalanced binary outcome data.

If identifying positives is more important than negatives (even though they are rarer), then need to strike "balance" between sensitivity and specificity. Having 90% sensitivity and only 40% specificity might be worth it depending on the problem.

Alleviating Severe Class Imbalance Problem

Can sometimes be difficult to conduct modelling or validate a model when there is severe class imbalance.

Two techniques to consider using for modelling:

- **Downsampling** - Randomly remove observations from the majority class until the majority and minority class are even (or close to even).
- **Upsampling** - Randomly duplicate observations from the minority class until the majority and minority class are even (or close to even).

Upsampling is usually better than downsampling, obviously because of the loss of information that occurs in the latter.

Applying either technique means we are skipping inference (since sample of data is no longer representative of the population). But doesn't compromise our prediction objective.

Nominal Categorical Outcome - Performance Measures

For nominal categorical outcomes, some binary performance measures generalize to this setting pretty well.

One approach is to use binary performance measures directly for each class.

Example Confusion Matrix for Three-Class Problem:

		Prediction		
		Cat	Dog	Fish
Truth	Cat	True Cat	False Dog	False Fish
	Dog	False Cat	True Dog	False Fish
	Fish	False Cat	False Dog	True Fish

Nominal Categorical Outcome - Binary Approach

If we're interested in first examining the cat class:

		Prediction		
		Cat	Dog	Fish
Truth	Cat	True Cat	False Dog	False Fish
	Dog	False Cat	True Dog	False Fish
	Fish	False Cat	False Dog	True Fish

Notice that the table illustrates the TP (True Cats), FP (False Cats), FN (False Dog and False Fish) and TN (remaining cells) with respect to the cat class.

"True negatives" are not differentiated (doesn't matter if it is actually a dog or fish, so long as it is correctly classified as **not** a cat).

Nominal Categorical Outcome - Toy Example

		Prediction		
		Cat	Dog	Fish
Truth	Cat	47	10	21
	Dog	19	45	26
	Fish	20	28	43

Sensitivity for Cat Class - $\frac{TP}{TP+FN} = \frac{47}{47+(10+21)} = 0.603$

Specificity for Cat Class - $\frac{TN}{TN+FP} = \frac{45+26+28+43}{(45+26+28+43)+(19+20)} = 0.785$

Sensitivity for Dog Class - $\frac{TP}{TP+FN} = \frac{45}{45+(19+26)} = 0.500$

Specificity for Dog Class - $\frac{TN}{TN+FP} = \frac{47+21+20+43}{(47+21+20+43)+(10+28)} = 0.775$

Nominal Categorical Outcome - Generalizations

Can therefore apply any of the binary performance measures to each of the classes of a nominal categorical outcome, with all other classes treated collectively as "negatives".

Referred to as the "One vs. Rest" or "One vs. All" approach.

Alternatively, if working with class probabilities, can also apply curve methodologies (ROC, P-R, etc.).

Example of predicted class probabilities:

Observation 1 - Cat=0.70, Dog=0.25, Fish=0.05

Can genereate an individual ROC or P-R curve for each class on the same plot (called multiclass ROC curves or multiclass P-R curves).

Nominal Categorical Outcome - Generalizations

Can also attempt to summarize overall performance by taking some aggregate of the individual binary performance measures (e.g., take a weighted average of the sensitivities).

Lastly, there are also performance measures for nominal categorical outcomes specifically (sometimes referred to as multiclass performance measures):

- Cohen's Kappa
- Matthew's multiclass correlation coefficient
- Log Loss/Cross-Entropy Loss (works with class probabilities)

Aggregate measures can get pretty difficult to interpret though, so might want to compute multiple different types of measures and assess/compare.

Ordinal Categorical Outcome - Performance Measures

A lot of the approaches discussed so far also generalize to ordinal categorical outcomes (i.e. just treat like multiple binary outcomes or a nominal categorical outcome.

Unfortunately, very little research performed so far to "take advantage" of ordinal structure.

Some examples of potential performance measures to use:

- Weighted Kappa
- Krippedorf's Alpha
- Somers' D

First two involve manually assigning weights to dictate penalization in differences. Last one involves counting number of concordant and discordant pairs.

Continous Outcome - Performance Measures

For continuous outcomes, quantifying performance is somewhat more straightforward, though there are still some elements to consider.

Two main questions to ask ourselves:

- Do we want to penalize predictions that are "way off" by an extra/exponential amount?
- Do we care about the absolute magnitude of the differences between the prediction and the truth, or do we only care about the relative magnitude of these differences?

Our choice will depend on the above, but we can always compute multiple performance measures as desired.

Continous Outcome - Performance Measures List

The following are the most common performance measures for continuous outcomes:

- Mean Squared Error (MSE)

$$\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

- Root Mean Squared Error (RMSE)

$$\sqrt{MSE}$$

- Mean Absolute Error (MAE):

$$\frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|$$

- Mean Absolute Percentage Error (MAPE):

$$100\% \times \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\hat{y}_i - y_i}{y_i}\right|$$

Continous Outcome - MSE and RMSE

MSE and RMSE will penalize predictions that are off by a lot very severely. Even if there are only a handful of predictions that are off by "a lot" and the rest are predicted perfectly, this error will be really high.

MSE and RMSE act very similarly, basically minimal difference. When comparing multiple models' RMSE, the ranking order will be the same as MSE but differences among lower error models will be reduced.

	Classifier			
	A	B	C	D
MSE	0.5	2	5	20
RMSE	0.707	1.414	2.236	4.472

Continous Outcome - MAE and MAPE

MAE and MAPE don't penalize large differences as much (note the use of absolute instead of square function for the differences).

MAE useful alternative if outliers "aren't a big deal". This is a measure with robustness to outlying predictions.

MAPE chosen when only the average relative difference between prediction and truth is of interest. Kind of like "normalizing" with respect to the magnitude of the true outcome.

Observations:

- Truth - 0.5 and 50
- Predictions - 0.4 and 45

MAE - $\frac{1}{2}(0.1 + 5) = 2.55$

MAPE - $\frac{1}{2}(\frac{0.1}{0.5} + \frac{5}{50}) \times 100\% = \frac{1}{2}(0.2 + 0.1) \times 100\% = 15\%$

Methods to Properly Assess Predictive Performance

As discussed in the machine learning workshop, we want to make sure that our assessment of predictive performance reflects what will occur in practice.

This means predictive performance should not be assessed on the training data (which was used to build the model to begin with), but instead on data that the model has not seen at all.

Main goal is to **estimate** generalization error (i.e. in general, how much prediction error will there be when using the model?). Multiple ways to do this.

Need to also consider how our hyperparameters for our model come into play during this.

Hold-Out Set Approach

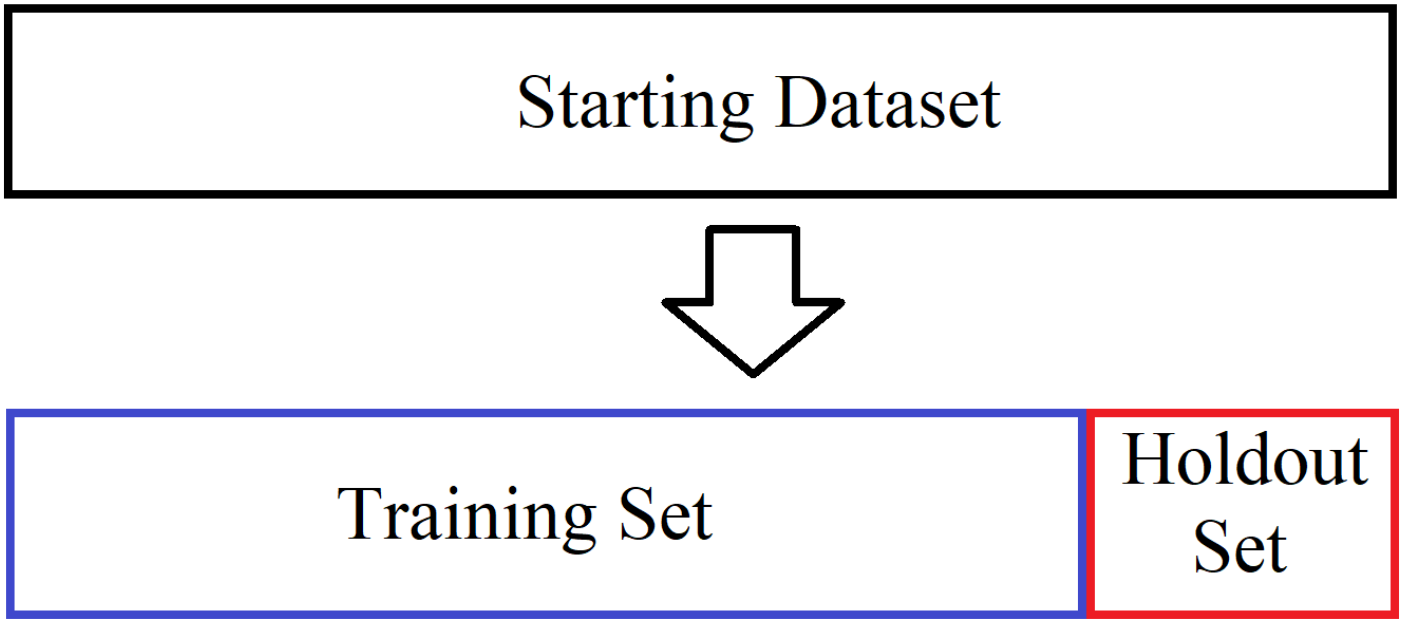
Absolute simplest method for predictive performance assessment is the holdout-set approach:

1. Split the dataset into training and test sets.
2. Train the model using the training set.
3. Compute predictive performance using the model on the held-out test set.

As you can see, the test set is not involved in the modelling at all, it only gets involved after the model is built.

Generally want around an 80%/20% split (so there is sufficient training data for the model).

Visualization of Hold-Out Approach



Discussion of Hold-Out Approach

Positive - Very simple to implement, and is not computationally intensive.

Negative - Requires removal of some data that could have been used for training process.

Too much data in the training set, can't properly assess predictive performance on small test set.

Too little data in the training set, can't model as effectively (need data to actually "learn").

Negative - Unstable measure of predictive performance (i.e. could have very high or low performance just by chance, since it is dependent on the contents of the hold-out set).

Cross-Validation Approach

Another approach to predictive performance assessment is the k-fold cross-validation:

1. Select the k you want to use (5 or 10 usually good choice)
2. Divide the complete dataset into k "folds" (of equal size)
3. Combine $k-1$ folds to form a training set and leave the remaining fold as a test set
4. Train the model using the training set
5. Compute predictions on the test set fold
6. Repeat Steps 3-5 for each different possible combination of k folds

Final result is predictions for each observation in the complete dataset, but only when they acted as test data.

Computing Predictive Performance from Cross-Validation Results

With the predictions on each observations, can conduct one of two options.

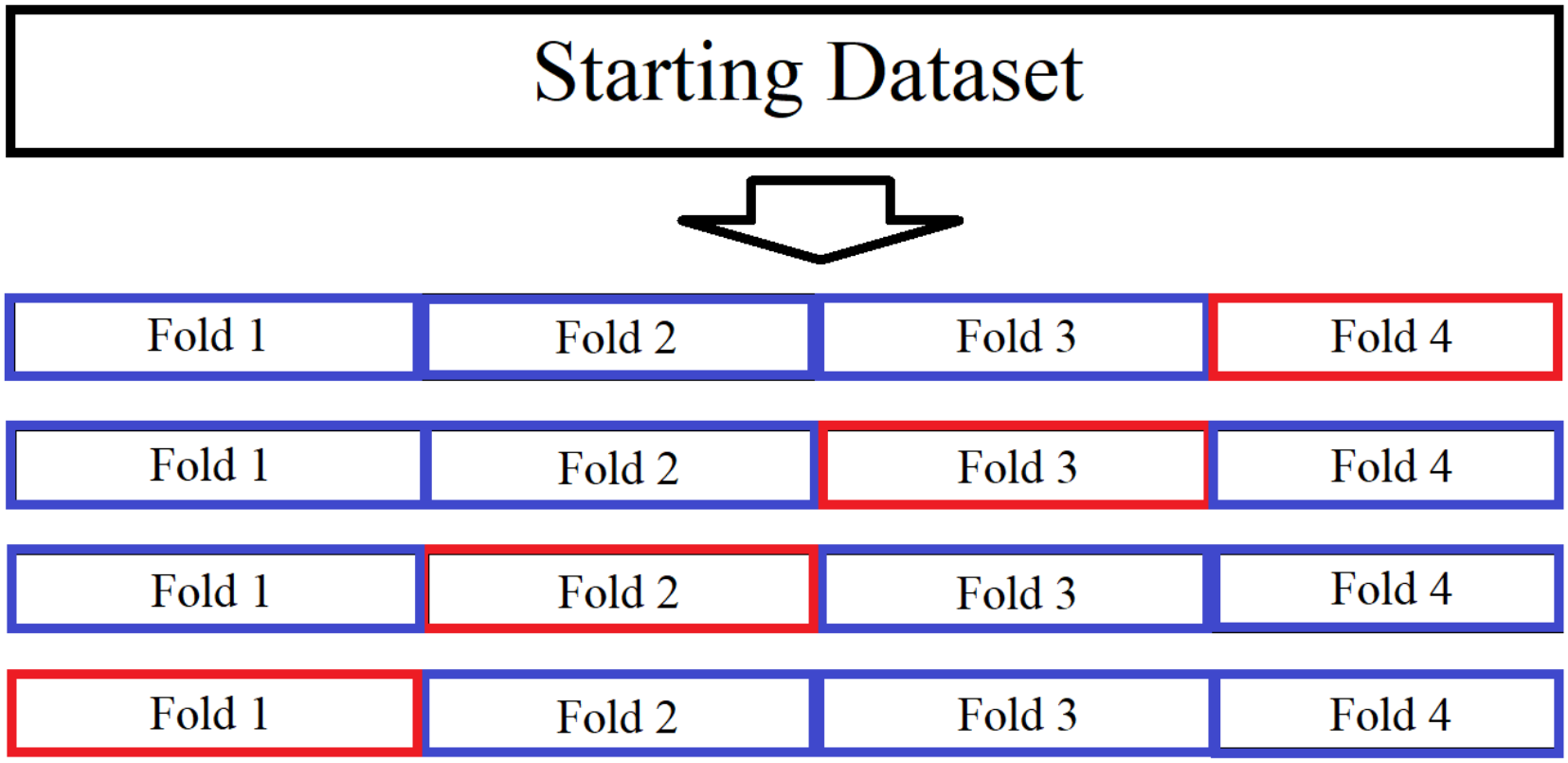
Option A:

- Compute the predictive performance measures on each test fold.
- Aggregate the performance measures, usually a (weighted) average, to construct a final performance measure.
- This option is the most common (especially for continuous outcomes).

Option B:

- Use all of the predictions to construct performance measure.
- Preferred option for performance curves, since aggregation of curves for each test fold doesn't really make sense.

Visualization of Cross-Validation Approach (4-fold)



Discussion of Cross-Validation Approach

Positive - Avoids complete removal of some data from the training process.

Positive - Similar to hold-out method, but runs over multiple iterations (making measure of predictive performance generally more stable).

Negative - Can be computationally intensive, especially with high k, since we need to build a model k times.

Negative - Validation pipelines can get really complicated when incorporating other elements (like hyperparameter tuning).

Extensions of Current Approaches

- Repeated cross-validation
- Leave-one-out cross-validation
- Stratified hold-out/cross-validation
- Hold-out/cross-validation with repeated measures
- Incorporation of Hyperparameter Tuning and Model Selection
  - Training/Validation/Test Approach
  - Nested cross-validation

Repeated Cross-Validation

Involves performing cross-validation multiple times (maybe 10s or 100s of times).

If we only do cross-validation once, there is still a concern that we may have split the data into the k folds too "conveniently", just by chance. This leads to performance instability still.

Reason to do repeated CV is to get even more stable estimates of predictive performance, since we are trying many different fold splits.

Can simply average performance measures across the repetitions to get final performance measure.

Leave-One-Out Cross-Validation

Leave-one-out cross-validation is essentially just n-fold cross-validation.

At each fold separation, all observations except one are used to train, and then we test on the removed observation.

Major advantage is that we only need to do leave-one-out cross-validation once per model (i.e. fold splits are obviously always the same). No need at all for repeated cross-validation.

Disadvantage is that we need to model n times. Very unrealistic to do for large datasets.

Stratified Hold-Out/Cross-Validation

With highly imbalanced (categorical) outcome data, it's possible for the test set or fold to not receive any observations from the rarer classes.

Can become very problematic to assess predictive performance effectively.

Need to stratify sets or folds to contain observations based on the proportion of each class present in the dataset.

Example:

1000 negatives + 100 positives $\rightarrow$ 10-fold cross-validation $\rightarrow$ 100 negatives and 10 positives for each fold (still randomly assigned).

Hold-Out/Cross-Validation with Repeated Measures

What happens when data is not completely independent (e.g., multiple observations from same patient)?

If we simply treat the data as independent, then the concern is that observations coming from the same cluster will show up in both the training and test set.

This is a problem, because if observations in the same cluster are correlated, then performance might be exaggerated because the trained model already has learned some information about observations in the test set (known as **data leakage**).

Solution: Need to divide dataset based on the clusters of observations (not the observations themselves).

How to Compute Predictive Performance with Hyperparameter Tuning or Model Selection?

We may also be interested in tuning hyperparameters for our machine learning method (e.g., k from k-Nearest Neighbours), or to conduct model selection (i.e. predictor selection).

Don't want to use performance on the training set to do this because then we'll overfit (nearly guaranteed).

Don't want to use performance on the test set because this causes **data leakage** (i.e. information from the test set helped train the model).

Need to use another independent dataset to help tune hyperparameters or choose optimal set of predictors.

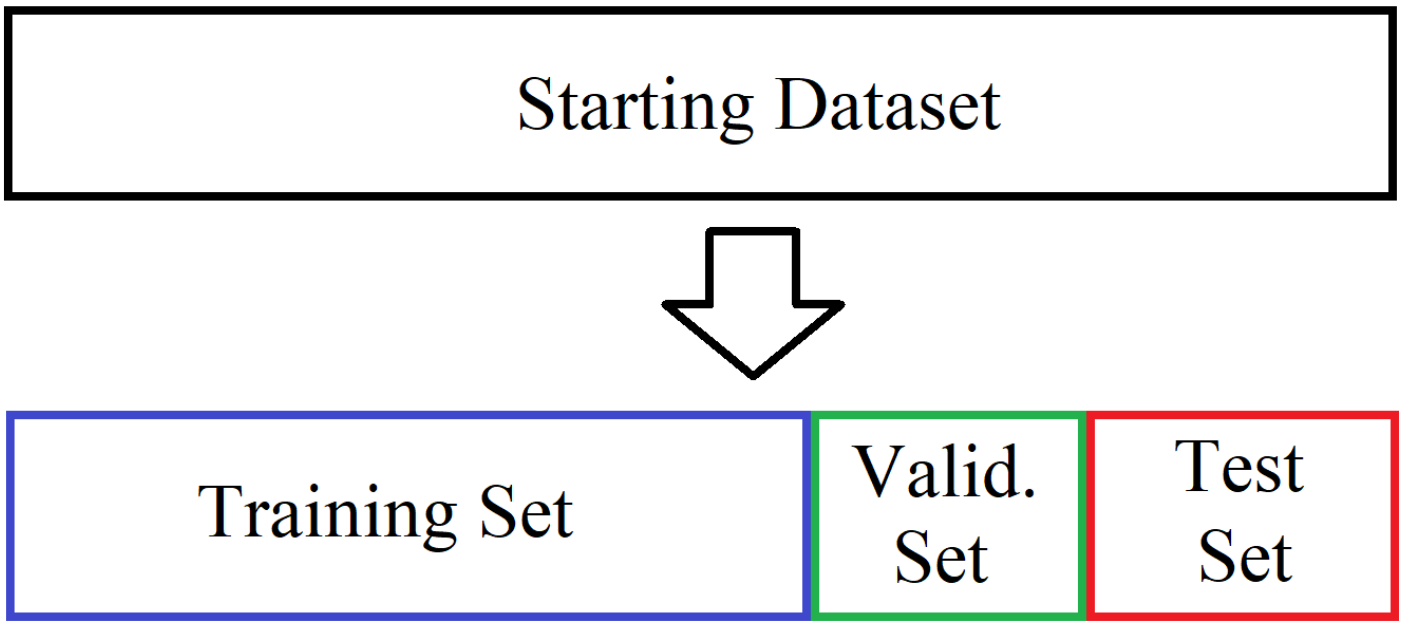
Hold-Out Extension for Hyperparameter Tuning and Model Selection

Proceed largely the same as before, just split the dataset even further:

1. Divide the dataset into three datasets: training set, validation set and test set.
2. Train the model using the training set.
3. Perform hyperparameter tuning or model selection by assessing performance of the model on the validation set.
4. Select the final hyperparameter value/model structure corresponding to the best performance achieved on the validation set.
5. Assess the selected model's predictive performance on the test set.

Generally want around an 80%/10%/10% split (so there is sufficient training data for the model).

Multiple Sets Visualization

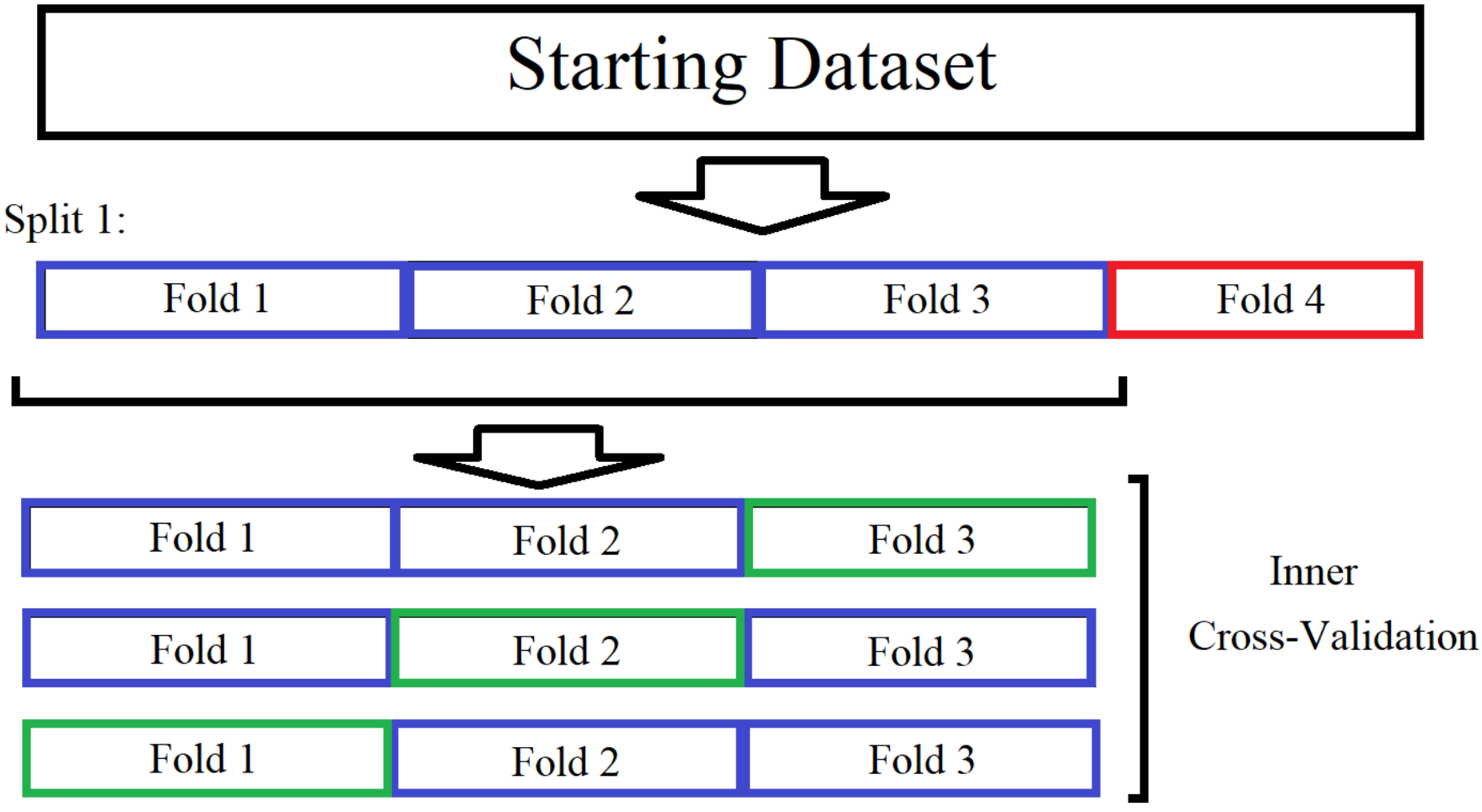


Cros-Validation Extension for Hyperparameter Tuning and Model Selection

Nested cross-validation involves an outer cross-validation loop, with another inner cross-validation loop nested inside (Outer k_1-fold cross-validation and Inner k_2-fold cross-validation).

1. Divide the complete dataset into k_1 "folds" (of equal size).
2. Combine $k_1 - 1$ folds to form a training set and leave the remaining fold as a test set.
3. **Divide the training set into k_2 folds (of equal size).**
4. **Conduct k_2 cross-validation on these folds, once for each hyperparameter/model type.**
5. **Select hyperparameter/model type that minimizes inner cross-validation predictive error.**
6. Train the model using the training set and selected hyperparameter/model type, then compute predictive performance on the test set.
7. Repeat Steps 2-6 for each different possible combination of the k_1 folds.

Nested Cross-Validation Visualization



Optimism-Corrected Bootstrap Validation

Some work has been done to investigate alternate ways to assess predictive performance, most notably involving bootstrapping.

Bootstrapping involves generating a "new" dataset that is the same size as the original, but by sampling observations with replacement from the original.

Other variants involving bootstrap sampling include the "0.632" and "0.632+" estimators[1], but focusing only on optimism-corrected bootstrap validation[2] here due to some prevalence in clinical and biostatistics research.

Main goal is to estimate the "optimism" in our training set performance estimate (i.e. how much are we over-exaggerating), and then correcting the performance estimate accordingly.

[1] See Efron (1983) and Efron and Tibshirani (1997), respectively

[2] See Harrell et al. (1996)

Optimism-Corrected Bootstrap Validation Procedure

General procedure is as follows:

1. Fit a model on the complete original dataset, compute "original" predictive performance (OPP).

2. Create bootstrapped dataset from original dataset.
3. Fit a new model to the bootstrapped dataset.
4. Use the new model to compute predictive performance on both the bootstrapped (BPP) and original (OBPP) datasets.
5. Compute and store the difference in performance (BPP-OBPP).

6. Repeat Steps 2-5 for as many bootstrap iterations desired.
7. Compute the average of (BPP-OBPP) differences, which gives the "optimism" (O).
8. Compute optimism corrected predictive performance using (OPP-O).

Potential Problems with Optimism-Corrected Bootstrap Validation

Note that this method doesn't derive its performance estimate by dividing the data into train/test splits (like in the hold-out and cross-validation approaches).

PROBLEM: There is data leakage, because most of the original dataset's observations are already found in the bootstrapped dataset.

So when we estimate OBPP, the original dataset is being used as a kind of "test" set, so we may overestimate OBPP.

Overestimating OBPP means underestimating optimism, which means overestimating the final predictive performance.

Optimism-Corrected Bootstrap Validation in Practice

Tends to be pretty accurate for low-dimensional dataset settings (i.e. ones with few predictor variables compared to the sample size).

Overestimates performance for high-dimensional datasets (sometimes by A LOT), particularly when there are more predictors than observations.

Refer to https://hbiostat.org/doc/simval.html for simulation by one of the authors of the method, indicating this performance issue themselves on logistic regression models.

Main Takeaway: Be cautious when using this method. Usually better to proceed with one of the other approaches discussed.

Calibration

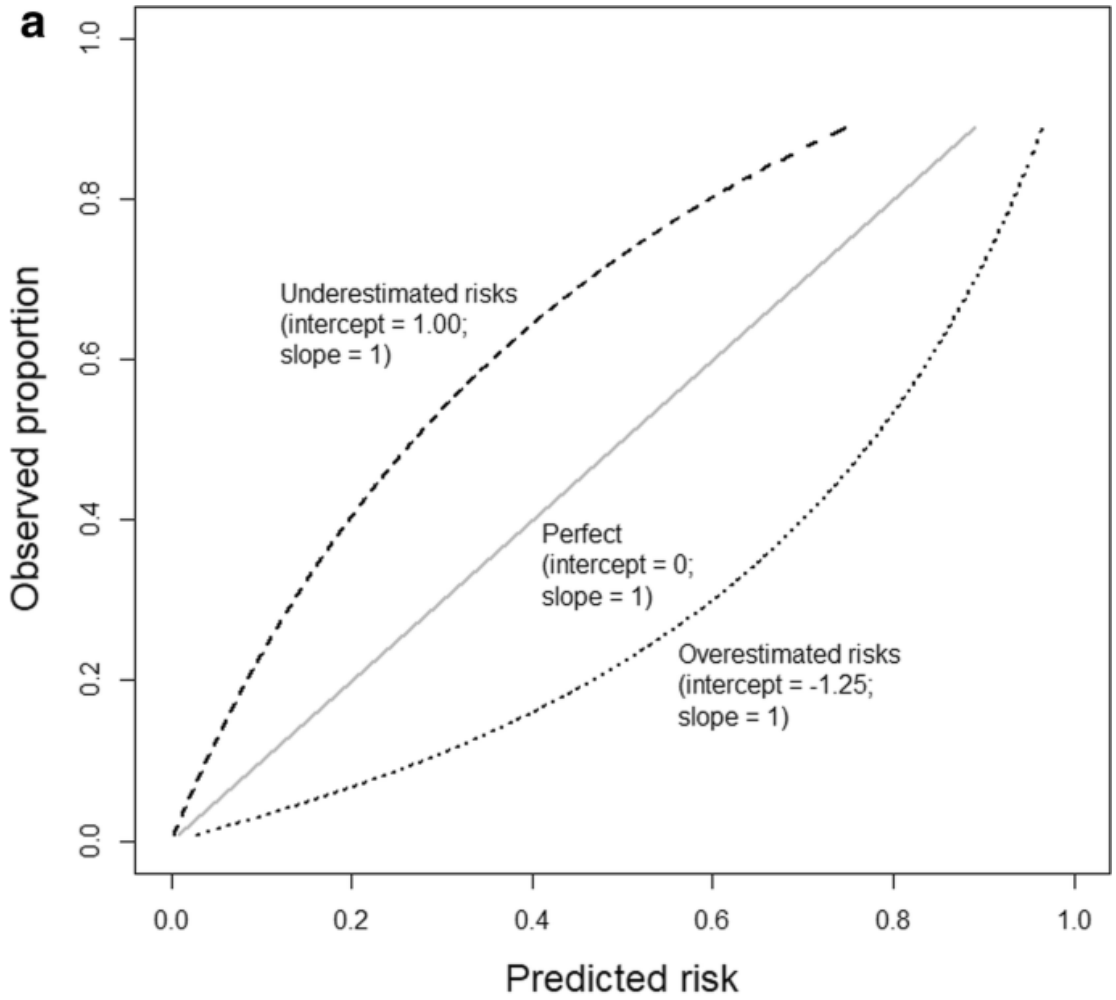
Another "type" of performance measure for binary outcomes. Different from the ones we discussed earlier, which involve the ability to distinguish between the classes.

Calibration mostly refers to the quality of the class probabilities output by the model.

Example: If a group of subjects with specific predictor values have an overall 20% risk of developing a condition, but the model assigns class probabilities of 40% instead to these individuals, then this is indicative of poor calibration.

Idea is to examine across the spectrum of actual class probabilities compared to the predicted class probabilities, do we get a straight line with intercept 0 and slope 1?

Calibration Curve Examples



[2]Figure is from Van Calster et al. (2019)

Calibration Correction

Calibration only really matters if you're interested in examining the class probabilities directly (which in some contexts is actually important).

If a model is identified to be poorly calibrated, you will need to proceed with correction procedures.

If only the calibration intercept is off, then often pretty straightforward for regression models, just correct the intercept accordingly.

If the calibration slope is off, then will require more complex updating of the class probabilities output by the model.

References

1. Efron, B. (1983). Estimating the error rate of a prediction rule: improvement on cross-validation. *Journal of the American statistical association*, *78*(382), 316-331.

2. Efron, B., & Tibshirani, R. (1997). Improvements on cross-validation: the 632+ bootstrap method. *Journal of the American Statistical Association*, *92*(438), 548-560.

3. Harrell Jr, F. E., Lee, K. L., & Mark, D. B. (1996). Multivariable prognostic models: issues in developing models, evaluating assumptions and adequacy, and measuring and reducing errors. *Statistics in Medicine*, *15*(4), 361-387.

4. Van Calster, B., McLernon, D. J., Van Smeden, M., Wynants, L., & Steyerberg, E. W. (2019). Calibration: the Achilles heel of predictive analytics. *BMC medicine*, *17*(1), 1-7.