Unit 4

Model assessment and selection

Model assessment and selection

- Model assessment aims at determining the *generalization* performance of the fitted model related to its predictive capability on independent test data.
  - This performance guides model selection.
  - One obtains a measure of quality for the ultimately chosen model.
- We have two separate goals:

  Model selection: estimating the performance of different models in order to choose the best one.

  Model assessment: having chosen a final model, estimating its prediction error (generalization error) on new data.

Bias, variance and model complexity

- Assume we have a quantitative target variable Y, a vector of inputs X and a prediction model $\hat{f}(X)$ that has been estimated from the training set $\mathcal{T}$.
- The loss function for measuring errors between Y and $\hat{f}(X)$ is denoted by $L(Y, \hat{f}(X))$, e.g.,

$$L(Y, \hat{f}(X)) = \begin{cases} (Y - \hat{f}(X))^2 & \text{squared error} \\ |Y - \hat{f}(X)| & \text{absolute error.} \end{cases}$$

- The *test error* (or *generalization error*) is the prediction error over an independent test sample:

$$\text{Err}_{\mathcal{T}} = \text{E}[L(Y, \hat{f}(X))|\mathcal{T}],$$

  where Y and X are drawn randomly from their joint distribution.

Bias, variance and model complexity / 2

- The expected prediction error (or expected test error) is given by

$$\mathrm{Err} = \mathrm{E}[L(Y, \hat{f}(X))] = \mathrm{E}[\mathrm{Err}_{\mathcal{T}}].$$

- Even if the goal is to estimate $\mathrm{Err}_{\mathcal{T}}$, sometimes an estimate for Err is easier to obtain.
- The *training error* is the average loss over the training sample:

$$\overline{\mathrm{err}} = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}(x_i)).$$

- The training error consistently decreases with model complexity.

Bias, variance and model complexity / 3

- Assume we have a qualitative target variable G taking one of K values in a set $\mathcal{G}$, labeled for convenience $1, \dots, K$.
- Typically we model the probabilities $p_k(X) = \Pr(G = k|X)$.
- Typical loss functions are:

$$L(G, \hat{G}(X)) = I(G \neq \hat{G}(X)),$$

$$L(G, \hat{p}(X)) = -\sum_{k=1}^{K} I(G = k) \log \hat{p}_k(X).$$

- Again the test error is

$$\mathrm{Err}_{\mathcal{T}} = \mathrm{E}[L(G, \hat{G}(X))|\mathcal{T}].$$

Bias, variance and model complexity / 4

- For general response densities the log-likelihood can be used as a loss function, e.g.,

$$L(Y, \theta(X)) = -2 \cdot \log \Pr_{\theta(X)}(Y).$$

Best approach

- In a data rich situation the best approach would be to split the data set into three parts:
  1. a training set
  2. a validation set
  3. a test set
- The *training set* is used to fit the models.
- The *validation set* is used to select the model.
- The *test set* is used to assess the generalization error of the final model.
- The optimal splitting of the number of observations into the three parts depends on the signal-to-noise ratio. A typical choice would be:

$$50\% – 25\% – 25\%$$

Bias-variance decomposition

- We assume

$$Y = f(X) + \epsilon,$$

where

$$E(\epsilon) = 0, \qquad \text{Var}(\epsilon) = \sigma_\epsilon^2.$$

- The expected prediction error is given by

$$\begin{aligned}
\text{Err}(x_0) &= E[(Y - \hat{f}(x_0))^2 | X = x_0] \\
&= \sigma_\epsilon^2 + [E\hat{f}(x_0)^2 - f(x_0)]^2 + E[\hat{f}(x_0) - E\hat{f}(x_0)]^2 \\
&= \sigma_\epsilon^2 + \text{Bias}^2(\hat{f}(x_0)) + \text{Var}(\hat{f}(x_0)) \\
&= \text{Irreducible Error} + \text{Bias}^2 + \text{Variance}.
\end{aligned}$$

Optimism of the training error rate

- Given a training set $\mathcal{T} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ the generalization error of a model $\hat{f}$ is

$$\text{Err}_{\mathcal{T}} = \text{E}_{X^0, Y^0}[L(Y^0, \hat{f}(X^0))|\mathcal{T}],$$

  where:
  - The training set $\mathcal{T}$ is fixed.
  - The point (X^0, Y^0) is a new test data point drawn from F, the joint distribution of the data.
- Averaging over training sets $\mathcal{T}$ yields the expected error

$$\text{Err} = \text{E}_{\mathcal{T}}\text{E}_{X^0, Y^0}[L(Y^0, \hat{f}(X^0))|\mathcal{T}].$$

Optimism of the training error rate / 2

- The training error is typically

$$\overline{\text{err}} = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}(x_i)).$$

- The *in-sample* error is

$$\text{Err}_{\text{in}} = \frac{1}{N}\sum_{i=1}^{N} \mathsf{E}_{Y^0}[L(Y^0, \hat{f}(x_i))|\mathcal{T}].$$

  where Y^0 denotes new responses for each of the N training points x_i, $i = 1, \ldots, N$.

- The *optimism* is given by

$$\text{op} \equiv \text{Err}_{\text{in}} - \overline{\text{err}}.$$

Optimism of the training error rate / 3

- The average optimism is obtained by taking the expectation of the optimism over training sets

$$\omega \equiv \mathsf{E}_{\boldsymbol{y}}(\mathrm{op}),$$

with the predictors in the training set being fixed.

- For squared error and 0–1 loss one can show

$$\omega = \frac{2}{N}\sum_{i=1}^{N}\mathrm{Cov}(\hat{y}_i, y_i).$$

- This implies

$$\mathsf{E}_{\boldsymbol{y}}(\mathrm{Err}_{\mathrm{in}}) = \mathsf{E}_{\boldsymbol{y}}(\overline{\mathrm{err}}) + \frac{2}{N}\sum_{i=1}^{N}\mathrm{Cov}(\hat{y}_i, y_i).$$

Optimism of the training error rate / 4

- Thus a prediction error based on the in-sample error can be estimated by estimating the optimism and adding it to the training error $\overline{\text{err}}$.
- In-sample error is in general not of direct interest, but still can be used for effective model selection. The reason is that the relative (rather than absolute) size of the error is what matters.

Estimates of in-sample prediction error

- The general form of the in-sample estimates is

$$\widehat{\mathrm{Err}}_{\mathrm{in}} = \overline{\mathrm{err}} + \hat{\omega},$$

where $\hat{\omega}$ is an estimate of the average optimism.

- If $\hat{y}_i$ is obtained by a linear fit with d inputs, the optimism simplifies to

$$\omega = \frac{2}{N} \sum_{i=1}^{N} \mathrm{Cov}(\hat{y}_i, y_i) = \frac{2}{N} d \sigma_\epsilon^2$$

for the additive error model $Y = f(X) + \epsilon$.

- When d parameters are fitted under the squared error loss, Mallows's C_p statistic is given by

$$C_p = \overline{\mathrm{err}} + 2 \cdot \frac{d}{N} \hat{\sigma}_\epsilon^2.$$

Estimates of in-sample prediction error / 2

- The *Akaike information criterion* also estimates Err_{in} but requires a log-likelihood loss function to be used:

$$-2 \cdot \mathsf{E}[\log \Pr_{\hat{\theta}}(Y)] \approx -\frac{2}{N}\mathsf{E}[\text{loglik}] + 2 \cdot \frac{d}{N},$$

  where
  - $\Pr_{\theta}(Y)$ is a family of densities for Y (containing the “true” density),
  - $\hat{\theta}$ is the maximum likelihood estimate and
  - “loglik” is the maximized log-likelihood.
- For the Gaussian model (with $\sigma_{\epsilon}^2 = \hat{\sigma}_{\epsilon}^2$ assumed known) C_p and AIC are equivalent.

Estimates of in-sample prediction error / 3

- Given a set of models $f_\alpha(x)$ indexed by a tuning parameter α, denote by $\overline{\text{err}}(\alpha)$ the training error and by $d(\alpha)$ the number of parameters for each model. Then one can define:

$$\text{AIC}(\alpha) = \overline{\text{err}}(\alpha) + 2 \cdot \frac{d(\alpha)}{N}\hat{\sigma}_\epsilon^2.$$

- The estimate $\hat{\sigma}_\epsilon^2$ is preferably estimated by a low-bias model.
- The function $AIC(\alpha)$ provides an estimate of the model performance, and we find the tuning parameter $\hat{\alpha}$ that minimizes it.
- An alternative implementation of the AIC would also include σ^2 in the parameter vector and hence have $d(\alpha) + 1$ parameters.

The effective numbers of parameters

- Assume $\boldsymbol{y} = (y_1, \ldots, y_N)$ and $\hat{\boldsymbol{y}} = (\hat{y}_1, \ldots, \hat{y}_N)$.
- If a linear fitting method is used:

$$\hat{\boldsymbol{y}} = \boldsymbol{S}\boldsymbol{y},$$

  where $\boldsymbol{S}$ is an $N \times N$ matrix depending on the input vectors x_i, but not on y_i.
- Then the *effective number of parameters* is defined as

$$\text{df}(\boldsymbol{S}) = \text{trace}(\boldsymbol{S}).$$

The Bayesian approach and BIC

- The Bayesian information criterion (BIC) is also applicable when maximum likelihood estimation is performed.
- The generic form of the BIC is

$$\text{BIC} = -2 \cdot \text{loglik} + (\log N) \cdot d.$$

- The BIC statistic times 1/2 is also known as Schwarz criterion.
- The estimate $\hat{\sigma}^2_\epsilon$ is also preferably estimated by a low-bias model.
- An alternative implementation of the BIC would also include σ^2 in the parameter vector and hence have $d + 1$ parameters.

The Bayesian approach and BIC / 2

- Suppose we have a set of candidate models $\mathcal{M}_m$, $m = 1, \ldots, M$ and corresponding parameter values θ_m.
- Assuming the prior distributions $\text{Pr}(\theta_m|\mathcal{M}_m)$ for the parameters of each model $\mathcal{M}_m$, the posterior probability of a model is given by:

$$\begin{aligned}\text{Pr}(\mathcal{M}_m|\boldsymbol{Z}) &\propto \text{Pr}(\mathcal{M}_m)\text{Pr}(\boldsymbol{Z}|\mathcal{M}_m)\\ &\propto \text{Pr}(\mathcal{M}_m) \int \text{Pr}(\boldsymbol{Z}|\theta_m, \mathcal{M}_m)\text{Pr}(\theta_m|\mathcal{M}_m)d\theta_m,\end{aligned}$$

  where $\boldsymbol{Z}$ represents the training data $\{x_i, y_i\}_1^N$.
- To compare two models $\mathcal{M}_m$ and $\mathcal{M}_l$, the posterior odds are determined by

$$\frac{\text{Pr}(\mathcal{M}_m|\boldsymbol{Z})}{\text{Pr}(\mathcal{M}_l|\boldsymbol{Z})} = \frac{\text{Pr}(\mathcal{M}_m)}{\text{Pr}(\mathcal{M}_l)} \cdot \frac{\text{Pr}(\boldsymbol{Z}|\mathcal{M}_m)}{\text{Pr}(\boldsymbol{Z}|\mathcal{M}_l)}.$$

The Bayesian approach and BIC / 3

- It the odds are greater than 1, model *m* is preferred, otherwise model *l*.
- The second factor is called *Bayes factor*,

$$\text{BF}(\boldsymbol{Z}) = \frac{\Pr(\boldsymbol{Z}|\mathcal{M}_m)}{\Pr(\boldsymbol{Z}|\mathcal{M}_l)}.$$

- Typically one assumes that the prior over models is uniform, so that $\Pr(\mathcal{M}_m)$ is constant.
- Using the so-called Laplace approximation for the integral followed by some other simplifications gives

$$\log \Pr(\boldsymbol{Z}|\mathcal{M}_m) = \log \Pr(\boldsymbol{Z}|\hat{\theta}, \mathcal{M}_m) - \frac{d_m}{2} \cdot \log N + O(1),$$

  where $\hat{\theta}$ is the ML estimate and d_m the number of free parameters in model $\mathcal{M}_m$.

The Bayesian approach and BIC / 4

- This implies that the posterior probability of each model $\mathcal{M}_m$ is given by

$$\frac{e^{-\frac{1}{2}\cdot \text{BIC}_m}}{\sum_{l=1}^{M} e^{-\frac{1}{2}\cdot \text{BIC}_l}}.$$

- The BIC is asymptotically consistent, i.e., the probability that it chooses the correct model if the true model is among the set of considered models approaches 1 for $N \to \infty$.
- The AIC tends to choose models which are too complex for $N \to \infty$.
- For finite samples, BIC often chooses models that are too simple, because of its heavy penalty on complexity.

Cross-validation

- Estimates the expected prediction error

$$\text{Err} = \text{E}[L(Y, \hat{f}(X))]$$

- If there is not enough data to split into training and validation, K-fold cross validation can be used:
  - The data is split into K roughly equal-sized parts.
  - For the kth part fit the model to the $K - 1$ parts of the data.
  - Calculate the prediction error of the fitted model when predicting the kth part of the data.
  - This is done for $k = 1, \dots, K$.
  - Combine the K estimates of prediction error.

Cross-validation / 2

- Let

$$\kappa : \{1, \ldots, N\} \to \{1, \ldots, K\}$$

  be an indexing function indicating the partition to which observation i is allocated by the randomization.
- Denote by $\hat{f}^{-k}(x)$ the fitted function, computed with the kth part of the data removed.
- Then the cross-validation estimate of the prediction error is

$$\text{CV}(\hat{f}) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{f}^{-\kappa(i)}(x_i)).$$

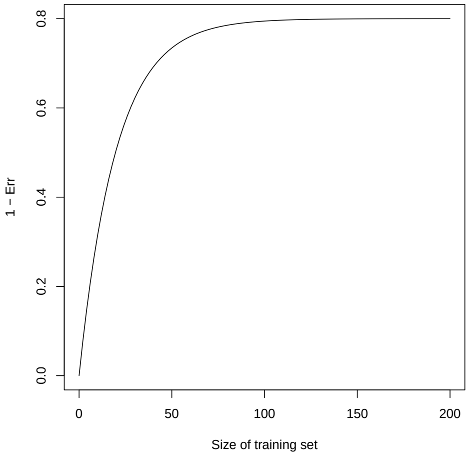
- Typical choices for K are 5 or 10.
- The case $K = N$ is known as *leave-one-out* cross-validation.

Cross-validation / 3

- The selection of a suitable K is again influenced by a variance-bias trade-off:
  - Small values of K:
    - high bias (only a much smaller data set used);
    - low variance (training sets differ).
  - Large values of K:
    - low bias (nearly data set of the same size used);
    - high variance (training sets overlap heavily);
    - often high computational burden.

Cross-validation / 4

Hypothetical learning curve



How to do cross-validation

- Consider a classification problem with a large number of predictors.
- A typical strategy might be:
  1. Screen the predictors: find a subset of “good” predictors that show a fairly strong (univariate) association with the class labels.
  2. Using just these predictors, build a multivariate classifier.
  3. Use cross-validation to estimate the unknown tuning parameters and to estimate the prediction error of the final model.
- This strategy is flawed because the pre-screening uses already the complete data.
- Cross-validation needs to be applied to the entire sequence of modeling steps.

How to do cross-validation / 2

- The correct way:
  1. Divide the samples into K cross-validation folds (groups) at random.
  2. For each fold $k = 1, 2, \ldots, K$:
     1. Find a subset of “good” predictors that show fairly strong (univariate) correlation with the class labels, using all of the samples except for fold k.
     2. Using just this subset of predictors, build a multivariate classifier, using all of the samples except those in fold k.
     3. Use the classifier to predict the class labels for samples in fold k.
  3. Accumulate the error estimates from the previous step over all K folds to produce the cross-validation estimate of prediction error.

Example: Prostate Cancer

```
> data("prostate", package = "ElemStatLearn")
> prostate.test <- prostate |>
+   dplyr::filter(!train) |>
+   dplyr::select(-train)
> prostate <- prostate |>
+   dplyr::filter(train) |>
+   dplyr::select(-train)
```

```
> formula <- lpsa ~ .
> mf <- model.frame(formula, data = prostate)
> y <- model.response(mf)
> X <- model.matrix(formula, mf)[, -1]
```

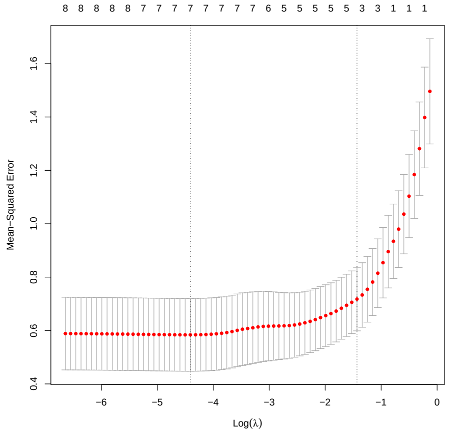
Example: Prostate Cancer / 2

```
> library("glmnet")
> cvlm <- cv.glmnet(X, y)
> cvlm
Call:  cv.glmnet(x = X, y = y)

Measure: Mean-Squared Error

    Lambda Index Measure    SE Nonzero
min 0.0122    47   0.584 0.136       7
1se 0.2389    15   0.718 0.119       3
```

Example: Prostate Cancer / 3



Example: Prostate Cancer / 4

```
> cbind(OLS = coef(cvlm, s = 0)[, 1],
+       lambda.min = coef(cvlm, s = "lambda.min")[, 1],
+       lambda.1se = coef(cvlm, s = "lambda.1se")[, 1],
+       Null = coef(cvlm, s = Inf)[, 1])
                   OLS lambda.min lambda.1se   Null
(Intercept)  0.3705042   0.172708    0.56383 2.4523
lcavol       0.5728285   0.546616    0.44259 0.0000
lweight      0.6132401   0.597821    0.34937 0.0000
age         -0.0186958  -0.015394    0.00000 0.0000
lbph         0.1437839   0.135706    0.00000 0.0000
svi          0.7312843   0.675719    0.18009 0.0000
lcp         -0.1999310  -0.150165    0.00000 0.0000
gleason     -0.0209603   0.000000    0.00000 0.0000
pgg45        0.0091561   0.007516    0.00000 0.0000
```

Example: Prostate Cancer / 5

```
> mf.test <- model.frame(formula, data = prostate.test)
> y.test <- model.response(mf.test)
> X.test <- model.matrix(formula, mf.test)[,-1]
> pred <- cbind(OLS = predict(cvlm, newx = X.test,
+                               s = 0)[,1],
+               predict(cvlm, newx = X.test,
+                         s = "lambda.min"),
+               predict(cvlm, newx = X.test),
+               Null = predict(cvlm, newx = X.test,
+                                s = Inf)[,1])
> head(pred, n = 3L)
      OLS lambda.min lambda.1se   Null
7  1.9662     1.9587     2.1036 2.4523
9  1.1696     1.1608     1.4568 2.4523
10 1.2622     1.2846     1.7961 2.4523
```

Example: Prostate Cancer / 6

```
> colMeans(sweep(pred, 1, y.test, "-")^2)
     OLS lambda.min lambda.1se       Null
 0.51724    0.49508    0.49654    1.05673
```

Bootstrap methods

- The bootstrap is a general tool to assess statistical accuracy.
- The bootstrap can be used to estimate extra-sample prediction error.
- The bootstrap seeks to estimate the conditional error $\text{Err}_{\mathcal{T}}$, but typically only estimates well the expected prediction error Err.

Bootstrap algorithm

- Denote the training set by $\boldsymbol{Z} = (z_1, z_2, \ldots, z_N)$ where $z_i = (x_i, y_i)$.
- Randomly draw datasets with replacement from the training data, each sample the same size as the original training set.
- This is done B times (e.g., $B = 100$), producing B bootstrap datasets.
- Let $S(\boldsymbol{Z})$ be any quantity computed from the data $\boldsymbol{Z}$, for example the prediction at some input point.

Bootstrap algorithm / 2

- Based on bootstrap sampling any aspect of the distribution of $S(\boldsymbol{Z})$ can be computed, for example its variance by

$$\widehat{\text{Var}} = \frac{1}{B-1} \sum_{b=1}^{B} (S(\boldsymbol{Z}^{*b}) - \bar{S}^*)^2,$$

where

$$\bar{S}^* = \frac{1}{B} \sum_{b=1}^{B} S(\boldsymbol{Z}^{*b}).$$

- $\widehat{\text{Var}}$ can be thought of as a Monte-Carlo estimate of the variance of $S(\boldsymbol{Z})$ under sampling from the empirical distribution $\hat{F}$ for the data $(z_1, z_2, \ldots, z_N)$.

Bootstrap algorithm / 3

- The prediction error can be estimated by

$$\widehat{\mathrm{Err}}_{\mathrm{boot}} = \frac{1}{B}\frac{1}{N}\sum_{b=1}^{B}\sum_{i=1}^{N} L(y_i, \hat{f}^{*b}(x_i)),$$

  where $\hat{f}^{*b}(x_i)$ is the predicted value at x_i.
- In this procedure the bootstrap samples are used as training data, while the observed data serves as test sample.
- As these sets have observations in common, overfit predictions might look unrealistically good.

Bootstrap algorithm / 4

- The probability that an observation is in the bootstrap sample is given by

$$\begin{aligned}\Pr(\text{observation } i \in \text{ bootstrap sample } b) &= 1 - \left(1 - \frac{1}{N}\right)^N \\ &\approx 1 - e^{-1} = 0.632.\end{aligned}$$

- A better estimate is obtained by keeping track of predictions from bootstrap samples not containing a specific observation.

Bootstrap algorithm / 5

- The leave-one-out bootstrap estimate of prediction error is defined by

$$\widehat{\text{Err}}^{(1)} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{|C^{-i}|} \sum_{b \in C^{-i}} L(y_i, \hat{f}^{*b}(x_i)),$$

  where C^{-i} is the set of bootstrap samples b that do *not* contain observation i. $|C^{-i}|$ is the number of these sets.
- Either B needs to be large enough to have $|C^{-i}| > 0$ or those indices need to be left out.
- The leave-one-out bootstrap solves the overfitting problem, but suffers from the training-set-size bias.

Bootstrap algorithm / 6

- The average number of distinct observations in each bootstrap sample is about

$$0.632 \cdot N.$$

  So the bias will be about the same as 2-fold cross-validation.

Evaluation of learners for a binary outcome

Performance characteristics

- The learner may either provide
  - a binary classification;
  - a numeric score, e.g., an estimate of the probability of the outcome being equal to 1.
- We might be interested to assess:
  - Discrimination
  - Calibration

Evaluation: Classification performance

The binary predictions of a classifier for binary outcomes are combined in a *confusion matrix*:

		predicted outcome		
		p	**n**	
actual outcome	$\mathbf{p}'$	True Positive (TP)	False Negative (FN)	P' = TP + FN
	$\mathbf{n}'$	False Positive (FP)	True Negative (TN)	N' = FP + TN
		P = TP + FP	N = FN + TN	

Evaluation: Classification performance / 2

- *Accuracy:*

$$\text{Acc} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

  The complement of the accuracy is the error rate or misclassification rate.
  Both measures are sensitive to imbalanced data.

- *Sensitivity & specificity:*
  Sensitivity, true positive rate (TPR), hit rate, recall:

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

  Specificity, true negative rate (TNR), inverse recall:

$$\text{TNR} = \frac{\text{TN}}{\text{FP} + \text{TN}}$$

Evaluation: Classification performance / 3

- *False positive and false negative rate:*
  False positive rate (FPR), false alarm rate (FAR):
  $$\text{FPR} = 1 - \text{TNR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$
  False negative rate (FNR), miss rate:
  $$\text{FNR} = 1 - \text{TPR} = \frac{\text{FN}}{\text{FN} + \text{TP}}$$
- *F-measure, F_1-score:*
  represents the harmonic mean of precision and recall:
  $$F\text{-measure} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}$$
  The F-measure uses only three elements of the confusion matrix. Hence two classifiers with different TN values may have the same F-measure.

Receiver operating characteristics (ROC)

- The receiver operating characteristics (ROC) curve is a two-dimensional graph where
  - TPR is on the y-axis and
  - FPR is on the x-axis.
- For classifiers returning a continuous score, the ROC curve is generated by changing the threshold on the confidence score. Each threshold generates only one point in the ROC curve.
- Comparing different classifiers in the ROC curve is not easy. The Area under the ROC curve (AUC) metric is used to calculate the area under the ROC curve and provides a measure for comparing classifiers.
- The AUC score is always bounded between zero and one. An AUC lower than 0.5 implies the classifier performs worse than random guessing.

Evaluation: Probabilistic predictions

- A *scoring rule* provides a summary measure for the evaluation of probabilistic predictions or forecasts.
- Proper scoring rules imply that the expected score for an observation drawn from the distribution F is minimized if the probabilistic forecast is F.
- For strictly proper scoring rules the minimum is unique.

Evaluation: Probabilistic predictions / 2

- *Brier score*:

$$\text{BS} = \frac{1}{N}\sum_{i=1}^{N}(\hat{p}_i - y_i)^2,$$

  where $\hat{p}_i$ is the predicted probability and y_i is the outcome value with values in $\{0, 1\}$. This is a strictly proper scoring rule.

- *Logarithmic scoring rule*:

$$\text{LS} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{p}_i) + (1 - y_i)\log(1 - \hat{p}_i).$$

  This is a *local* strictly proper scoring rule. A scoring rule is local if the probabilistic forecast is evaluated only at the actual outcome.

Evaluation: Probabilistic predictions / 3

- The probability of concordance, *c*, between predicted probability and response is derived from the Wilcoxon-Mann-Whitney two-sample rank test. The *c* index is identical to the AUC score.
- Somers' D_{xy} rank correlation between predicted probability and observed responses is related to *c* by the identity:

  $$D_{xy} = 2(c - 0.5),$$

  with D_{xy} being the difference between concordance and discordance probabilities.

Example: South African Heart Disease

```
> set.seed(123)
> train <- sample(nrow(SAheart), 0.75 * nrow(SAheart))
> model1 <- glm(chd ~ age, data = SAheart[train,],
+               family = binomial())
> actual <- SAheart$chd[-train]
> score1 <- predict(model1, type = "response",
+                   newdata = SAheart[-train,])
> pred1 <- as.integer(score1 > 0.5)
```

Example: South African Heart Disease / 2

```
> library("caret")
> confusionMatrix(table(pred1, actual))
Confusion Matrix and Statistics

     actual
pred1  0  1
    0 63 24
    1 12 17

               Accuracy : 0.69
                 95% CI : (0.597, 0.772)
    No Information Rate : 0.647
    P-Value [Acc > NIR] : 0.1917

                  Kappa : 0.273
```

Example: South African Heart Disease / 3

```
Mcnemar's Test P-Value : 0.0668

           Sensitivity : 0.840
           Specificity : 0.415
        Pos Pred Value : 0.724
        Neg Pred Value : 0.586
            Prevalence : 0.647
        Detection Rate : 0.543
  Detection Prevalence : 0.750
     Balanced Accuracy : 0.627

      'Positive' Class : 0
```

Example: South African Heart Disease / 4

```
> model2 <- glm(chd ~ ., data = SAheart[train,],
+               family = binomial())
> score2 <- predict(model2, type = "response",
+                   newdata = SAheart[-train,])
> pred2 <- as.integer(score2 > 0.5)
```

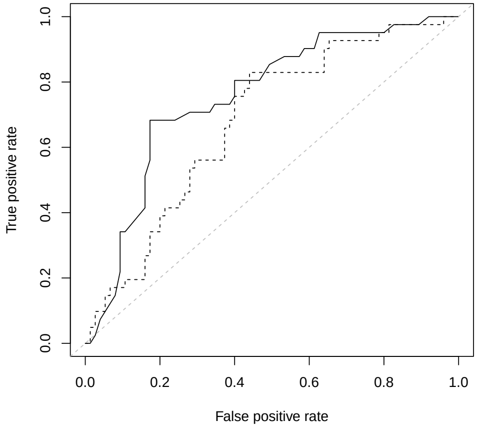
Example: South African Heart Disease / 5

```
> library("mlr")
> Brier <- c(age = measureBrier(score1, actual,
+                                  "0", "1"),
+            full = measureBrier(score2, actual,
+                                  "0", "1"))
> probs1 <- cbind(‘0‘ = 1 - score1, ‘1‘ = score1)
> probs2 <- cbind(‘0‘ = 1 - score2, ‘1‘ = score2)
> LS <- c(age = measureLogloss(probs1, actual),
+         full = measureLogloss(probs2, actual))
> cbind(Brier = Brier, LS = LS)
       Brier      LS
age  0.18858 0.55613
full 0.23347 0.67381
```

Example: South African Heart Disease / 6

```
> library("ROCR")
> predob1 <- prediction(score1, actual)
> perf1 <- performance(predob1 , "tpr", "fpr")
> plot(perf1)
> predob2 <- prediction(score2, actual)
> perf2 <- performance(predob2 , "tpr", "fpr")
> plot(perf2, add = TRUE, lty = 2)
> abline(a = 0, b = 1, col = "grey", lty = 2)
```

Example: South African Heart Disease / 7


True positive rate
False positive rate