Validation & model selection

Artificial Intelligence **3rd year INF**

Expected loss

- $\mathbf{X}, Y$ are random variables with the joint distribution $P(\mathbf{X}, Y)$
- Predictor: $h\colon \mathbf{x} \in \mathcal{X} \to h(\mathbf{x}) \in \mathcal{Y}$
- Loss function: $L(h(\mathbf{X}), Y)$
- **Expected loss:** $L[h] = \mathbb{E}[L(h(\mathbf{X}), Y)]$

  - In classification: $\mathbb{I}(h(\mathbf{X}) \neq Y) = \begin{cases} 1 \text{ si } & h(\mathbf{X}) \neq Y \\ 0 \text{ si } & h(\mathbf{X}) = Y \end{cases}$

Precision = 1 − Error

$$\text{Error} = \mathbb{E}[h(\mathbf{X}) \neq Y]$$

  - In regression: $|h(\mathbf{X}) - Y|^p$

MSE: mean squared error

$$\text{MSE} = \mathbb{E}\left[|h(\mathbf{X}) - Y|^2\right]$$

MAE:mean absolute error

$$MAE = \mathbb{E}[|h(\mathbf{X}) - Y|]$$

Learning from data

$\boldsymbol{\theta}$: Model parameters

- Learning algorithm $\mathcal{L}$: $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N} \rightarrow h(\cdot\,; \boldsymbol{\theta})$
- Set of labeled data $\mathcal{D}_{train} = \left\{\left(\mathbf{x}_n^{train}, y_n^{train}\right)\right\}_{n=1}^{N_{train}}$

Training instances

- **Expected loss**: $L(\boldsymbol{\theta}) = \mathbb{E}[L(h(\mathbf{X}\,; \boldsymbol{\theta}), Y)]$
- **Complexity penalty** (**regularization terms**): $\{R_i(\boldsymbol{\theta})\}_{i=1}^{I}$
- **Training**: Minimize the cost function estimated in $\mathcal{D}_{train}$

$$L_{train}(\boldsymbol{\theta}) = \frac{1}{N_{train}} \sum_{n=1}^{N_{train}} L\left(h\left(\mathbf{x}_n^{train}; \boldsymbol{\theta}\right), y_n^{train}\right)$$

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \left(L_{train}(\boldsymbol{\theta}) + \sum_{i=1}^{N} \lambda_i R_i(\boldsymbol{\theta})\right)$$

Regularization terms included to improve the generalization capacity of the predictive model.

- **Expected loss**: $L(\boldsymbol{\theta}^*) = \mathbb{E}[L(h(\mathbf{X}; \boldsymbol{\theta}^*), Y)]$

In general, $L_{train}(\boldsymbol{\theta}^*) < L(\boldsymbol{\theta}^*)$

Validating a predictor

- The **training estimate** of the expected loss is generally **biased**.

  Typically, $L_{train}(\boldsymbol{\theta}^*) < L(\boldsymbol{\theta}^*)$

- **Labeled test data**: $\mathcal{D}_{test} = \left\{\left(\mathbf{x}_n^{test}, y_n^{test}\right)\right\}_{n=1}^{N_{test}}$

  Test data **independent** of training data: $\mathcal{D}_{test} \perp \mathcal{D}_{train}$

$$L_{test}(\boldsymbol{\theta}^*) = \frac{1}{N_{test}} \sum_{n=1}^{N_{test}} L\left(h\left(\mathbf{x}_n^{test}; \boldsymbol{\theta}^*\right), y_n^{test}\right)$$

- The average loss in the **test data** is an **unbiased estimate** of the actual **expected loss**:

  $L(\boldsymbol{\theta}^*) = \mathbb{E}[L(h(\mathbf{X}; \boldsymbol{\theta}^*), Y)] \approx L_{test}(\boldsymbol{\theta}^*)$

Underfitting / overfitting

- **Underfitting**

  The type of predictor considered **has low expressive capacity.** In consequence, it is not able to capture the dependencies between the attributes and the variable to be predicted.

  The expected loss of the predictor is too large.

- **Overfitting**

  The type of predictor considered is **too flexible** and learns spurious patterns that are not relevant for prediction (e.g. sampling fluctuations, noise, outliers, etc.).

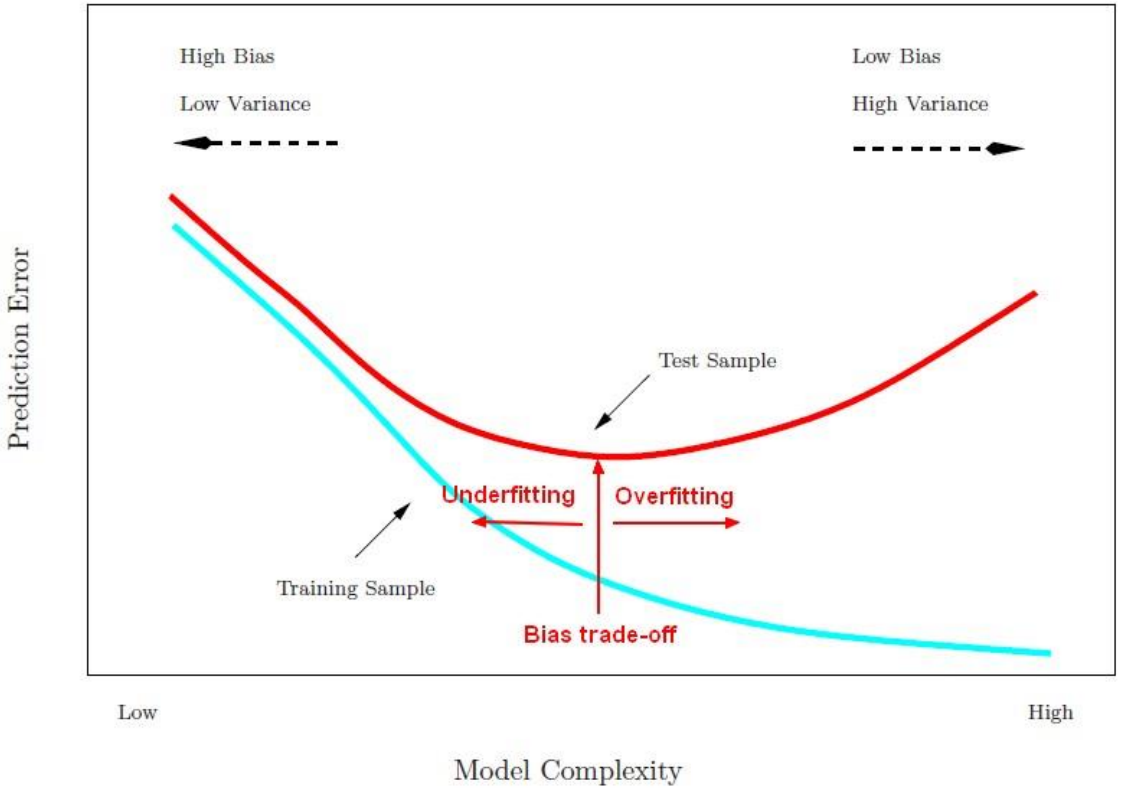
  Training estimate of the expected loss is too optimistic and underestimates the actual expected loss.

Expressive capacity of the model

- **Rigid models** (e.g. linear models)
  - **Low expressive capacity**: only simple dependencies
  - Prone to **underfitting**.
  - Generally have **high bias**, **low variance**.

- **Flexible models** (e.g. neural networks)
  - **High expressive capacity**: can model complex dependencies
  - Need to control **overfitting**
  - Generally have **low bias**, **high variance**.

Underfitting / overfitting



Source: https://gerardnico.com/ under

license CC Attribution-Noncommercial-Share Alike 4.0 International

Architecture / parameters / hyperparameters

- **Architecture / configuration:** Specification of the functional form of the predictor (e.g. number and type of hidden layers and neurons in a neural network, choice of kernel in an SVM, etc.).
- **Model parameters ($\boldsymbol{\theta}$):** Values needed to specify the prediction system (e.g. synaptic weights in a neural network, support vectors in an SVM, etc.). They are determined by training the model using labeled data data.
- **Configuration of the learning algorithm:**
  - Function to optimize (e.g. likelihood, posterior probability, Mean squared error, etc.).
  - Regularization terms: (e.g. L_1, L_2 penalties).
  - Optimization terms: quasi-Newton, gradient descent, with or without momentum
  - Type of optimization (e.g. batch / stochastic gradient descent (SGD) / online).
  - Hyperparameters:
    - Of the predictor (e.g. kernel width in SVM)
    - Of the cost function (e.g. relative strength of the regularization terms)
    - Of the learning algorithm: learning rate, strength of momentum term, size of mini-batches in SGD, etc.

Predictive models

$$h(\cdot;\boldsymbol{\theta}): \quad \mathcal{X} \rightarrow \mathcal{Y}$$

$$\mathbf{x} \in \mathcal{X} \rightarrow h(\mathbf{x};\boldsymbol{\theta}) \in \mathcal{Y}$$

Predictor	**Architecture / configuration**	**Model parameters (θ)**
Nearest neighbors	• Distance function (metric)	• Number of neighbors
Decision trees	• Type of tree (ID3, C4.5, CART) • Splitting criterion (Gini / entropy)	• Depth of the tree, • Minimum number of instances in nodes for split
Neural network (MLP)	• # hidden layers • # nodes in the hidden layer • Transfer function (sigmoidal, tanh, ReLu, etc.)	• Synaptic weights
Support Vector Machine (SVM)	• Type of kernel (linear, polynomic, Gaussian)	• Kernel parameters (usually inverse width γ) • Strength of regularization (C) • In SVR: Noise level (ε)

Error measures: Classification

Given labeled data $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$

and a classifier $h\colon \mathbf{x} \in \mathcal{X} \rightarrow h(\mathbf{x}) \in \mathcal{Y}$

- Classification error

$$\text{Error} = \mathbb{E}[\mathbb{I}(h(\mathbf{X};\boldsymbol{\theta}) \neq Y)] \approx \frac{1}{N}\sum_{n=1}^{N} \mathbb{I}[(h(\mathbf{x}_n;\boldsymbol{\theta}) \neq y_n)]$$

- Confusion matrix: $y \in \{C_1, C_2, \ldots, C_K\}$

$(Conf)_{ij}$ = # instances of class C_j classified as C_i

Error = 24.20 %

		Actual class label		
		C_1	C_2	C_3
Predicted class label	C_1	170	10	47
	C_2	4	200	21
	C_3	70	15	153

Error measures: Regression

Given labeled data $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$ + predictor $h(\mathbf{x}; \boldsymbol{\theta}) \in \mathbb{R}$

- **Mean squared error**:

$$MSE = \mathbb{E}\left[(h(\mathbf{X}; \boldsymbol{\theta}) - Y)^2\right] \approx \frac{1}{N}\sum_{n=1}^N (h(\mathbf{x}_n; \boldsymbol{\theta}) - y_n)^2$$

- **Normalized mean-squared error**: $NMSE = \frac{MSE}{Var}$;

$$Var = \frac{1}{N}\sum_{n=1}^N (y_n - \bar{y})^2 ; \qquad \bar{y} = \frac{1}{N}\sum_{n=1}^N y_n$$

- **Mean absolute error**:

$$MAE = \mathbb{E}[|h(\mathbf{X}; \boldsymbol{\theta}) - Y|] \approx \frac{1}{N}\sum_{n=1}^N |h(\mathbf{x}_n; \boldsymbol{\theta}) - y_n|$$

- **Minimax error**: $MME = \max_{1 \le n \le N} |h(\mathbf{x}_n; \boldsymbol{\theta}) - y_n|$

No free lunch (NFL) theorems

The **accuracy** of any given predictor **averaged over all** possible **classification problems** is equal to that of **random guessing**.

Corollary: Positive performance in some learning situations must be offset by a negative performance in others.

- Schaffer, C. (1994). “A conservation law for generalization performance”. In *Machine Learning: Proceedings of the Eleventh International Conference*, Cohen and Hirsh, eds. Morgan Kaufmann, San Mateo, CA.
- Wolpert, David (1996), "The Lack of A Priori Distinctions between Learning Algorithms", Neural Computation, pp. 1341-1390.

Keys to successful classification: Domain knowledge, inductive biases, diversification

- To be **effective learning** must make use of **domain knowledge**
  - **Feature selection**
  - **Feature construction**
  - Take advantage of the **structure** of the problem
- Need to use **inductive biases**. In general, prefer "simple" models
  - **Regularization terms** in the cost function
  - Priors in a **Bayesian** setting (bonus: provides **uncertainties**)
- Use **diversification** mechanisms
  - **Ensembles**: Subsampling (bagging), random features (random forest), noise in class labels (class-switching ensembles), etc.
  - **Randomization** steps in the optimization (stochastic gradient descent, dropout in deep neural networks, etc.)

Occam's razor (bias for induction)

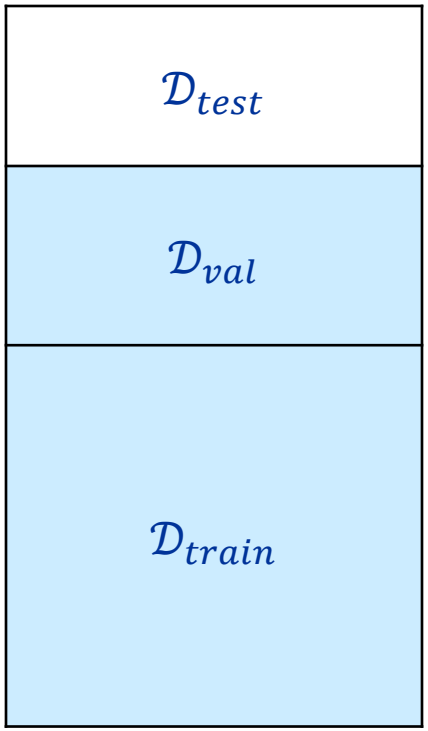
GOAL: Find the **optimal balance** between **expressive capacity** and **robustness** of the predictions (e.g. use regularization terms, use appropriate priors in Bayesian approach, etc.). This balance depends on the **type of problem** and the **amount of data**.

- **Recommended procedure**:
  - **Start with a simple model** (e.g. linear) and use it as a baseline.
  - Try a **sequence of progressively more complex** (more expressive) models.
  - **Monitor the quality of the predictions** (e.g. the accuracy) using validation techniques that avoid biases.
  - Select the model that has the **lowest complexity** and has a **sufficiently high** value of the monitored **quality** measure.

Validation set (holdout)

Partition the labeled data $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$ at random (*)

$$\mathcal{D} = [\mathcal{D}_{train} \cup \mathcal{D}_{val}] \cup \mathcal{D}_{test}$$



* Build predictor with $\mathcal{D}_{train}$

$$\text{Err}_{train} < \text{Err}_{val} \approx \text{Err}_{test}$$

* Build predictor with $[\mathcal{D}_{train} \cup \mathcal{D}_{val}]$

$\mathcal{D}_{train}$: Minimize cost function

$\mathcal{D}_{val}$: Determine architecture, conf., hyperparameters, etc.

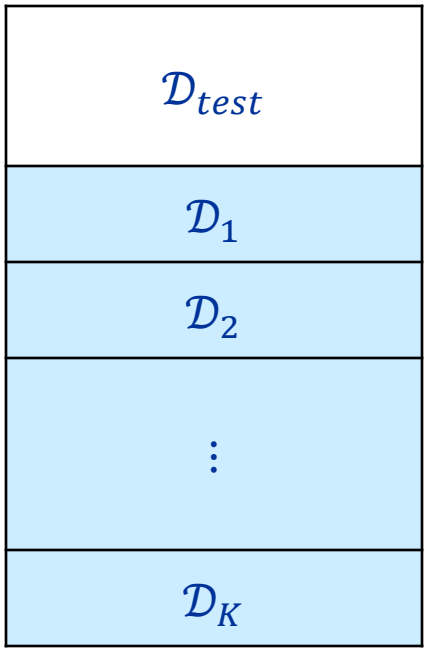
$$\text{Err}_{train} < \text{Err}_{val} < \text{Err}_{test}$$

(*) **Stratification** may be useful (maintain original class percentages in each of the sets into which $\mathcal{D}$ is partitioned)

K-fold cross validation

Typically K = 10
10-fold CV

Partition the labeled data $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$ at random



$$\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{test} \ (*)$$

$$\mathcal{D}_{train} = \mathcal{D}_1 \cup \mathcal{D}_2 \cup \ \ldots \ \cup \mathcal{D}_K \ \text{[K folds]} \ (*)$$

$$\mathcal{D}_i \cap \mathcal{D}_j = \emptyset \quad \text{for } i \neq j$$

for k := 1 To K

$$\mathcal{D}_{train}^{[k]} = \bigcup_{\substack{i=1 \\ i \neq k}}^{K} \mathcal{D}_i \ \Rightarrow \qquad h\left(\mathbf{x};\ \boldsymbol{\theta}_k^*\right)$$

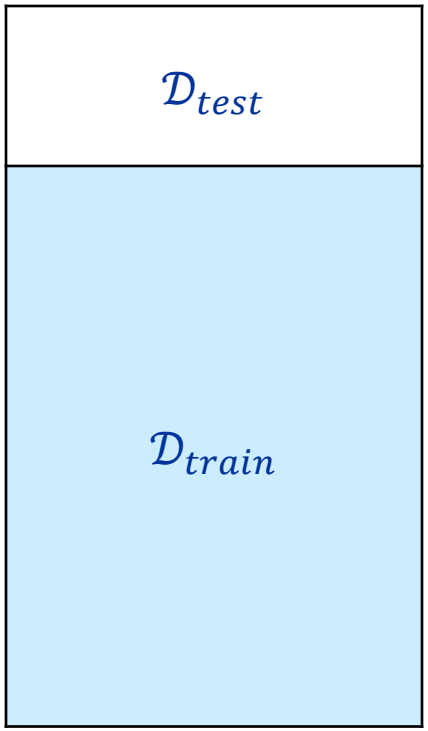
$$\text{Err}_k = \text{error rate of } h\left(\mathbf{x};\ \boldsymbol{\theta}_k^*\right) \text{ on } \mathcal{D}_{train}^{[k]}$$

$$\text{Err}_{K-CV} = \frac{1}{K}\sum_{k=1}^{K} \text{Err}_k; \quad \text{stdev}_{K-CV}$$

(*) **Stratification** may be useful

Leave-one-out estimates ($K = N_{train}$)

Partition the labeled data $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$ at random



$$\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{test} \; (*)$$

$$\mathcal{D}_{train} = \left\{\left(\mathbf{x}_n^{train}, y_n^{train}\right)\right\}_{n=1}^{N_{train}}$$

for n := 1 To N_{train}

$$\mathcal{D}_{train}^{[n]} = \left\{\left(\mathbf{x}_m^{train}, y_m^{train}\right)\right\}_{\substack{m=1; \\ m \neq n}}^{N_{train}} \Rightarrow h(\mathbf{x};\, \boldsymbol{\theta}_n^*)$$

$$\mathrm{Err}_n = \mathrm{Err \ of} \; h(\mathbf{x};\, \boldsymbol{\theta}_n^*) \; \mathrm{on} \left(\mathbf{x}_n^{train}, y_n^{train}\right)$$

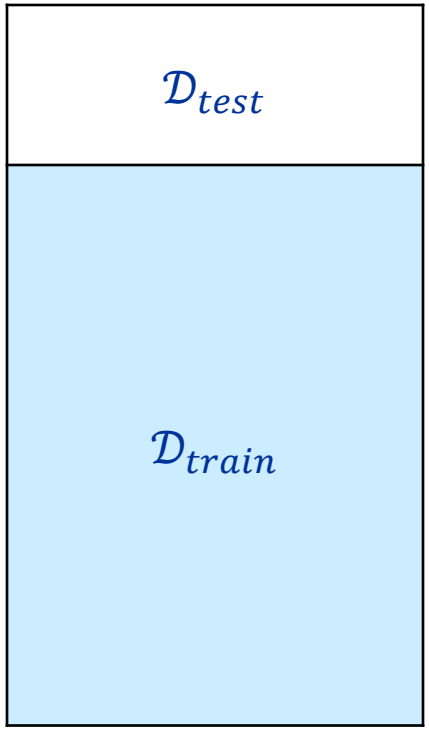
$$\mathrm{Err}_{loo} = \frac{1}{N_{train}} \sum_{n=1}^{N_{train}} \mathrm{Err}_n$$

Typically, **low bias & high variance**

(*) **Stratification** may be useful

Bootstrap estimates

Partition the labeled data $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$ at random



$\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{test}$

Resampled with or without repetition from the indices of the training instances $\{n\}_{n=1}^{N_{train}}$

for b := 1 To B

Bootstrap sample(*): $\{n_m\}_{m=1}^{N_{bag}}$

$\mathcal{D}_{train}^{[b]} = \left\{\left(\mathbf{x}_{n_m}^{train}, y_{n_m}^{train}\right)\right\}_{m=1}^{N_{bag}} \Rightarrow h\left(\mathbf{x};\ \boldsymbol{\theta}_b^*\right)$

$\mathrm{Err}_b = \mathrm{Err}$ of $h\left(\mathbf{x};\ \boldsymbol{\theta}_b^*\right)$ on $\mathcal{D}_{train} \setminus \mathcal{D}_{train}^{[b]}$

Out of bag error

$\mathrm{Err}_{boot} = \frac{1}{B}\sum_{b=1}^{B} \mathrm{Err}_b$

(*) Typically, with repetition and $N_{bag} = N_{train}$ (**63.21 % of training instances**)

Estimates of the generalization error

	Bias (Typically,)	**Variance (Typically,)**
Err_{train}	$\mathrm{Err}_{train} < \mathrm{Err}_{test}$ (optimistic)	Low
Err_{boot}	$\mathrm{Err}_{boot} > \mathrm{Err}_{test}$ (pesimistic)	
Err_{2-CV}	$\mathrm{Err}_{2-CV} > \mathrm{Err}_{test}$ (pesimistic)	High
Err_{10-CV}	$\mathrm{Err}_{10-CV} \gtrsim \mathrm{Err}_{test}$ (pesimistic)	
Err_{loo}	$\mathrm{Err}_{loo} \approx \mathrm{Err}_{test}$ (unbiased)	Very high

Warning: No unbiased estimator of the variance!

Corrected estimates (compensate biases)

$$\mathrm{Err}_{0.632} = 0.368\,\mathrm{Err}_{train} + 0.632\,\mathrm{Err}_{boot}$$

$$\mathrm{Err}_{0.632+} = (1-w)\,\mathrm{Err}_{train} + w\,\mathrm{Err}_{boot}$$

$$w = \frac{0.632}{1-0.368\,R}; \quad \mathrm{R} = \frac{\mathrm{Err}_{boot}-\mathrm{Err}_{train}}{\gamma-\mathrm{Err}_{train}};$$

Corrects optimistic bias of $\mathrm{Err}_{0.632}$ when Err_{train} is too small (overfitting)

$$\gamma = \frac{1}{N_{train}^2}\sum_{n=1}^{N_{train}}\sum_{m=1}^{N_{train}} L\left(h\left(\mathbf{x}_n^{train};\boldsymbol{\theta}\right), y_m^{train}\right)$$

Common evaluation protocols

- 10 × 10-fold CV:

  $$\text{Average}(\text{Err}_{10-CV}) \pm \text{Average}(\text{stdev}_{10-CV})$$

- 100 × random train / test partitions

  $$\text{Average}(\text{Err}_{test}) \pm \text{Stdev}(\text{Err}_{test})$$

WARNING: No unbiased estimator of the variance!

The stdev computed cannot be used in standard tests to determine whether differences in error are statistically significant.

Corrected statistical tests (e.g. corrected resampled *t*-test):

- Claude Nadeau, Yoshua Bengio "Inference for the Generalization Error", Machine Learning 52 (3), 239-281 (2003) [https://doi.org/10.1023/A:1024068626366]
- Janez Demšar “Statistical Comparisons of Classifiers over Multiple Data Sets” J. Mach. Learn. Res. 7, 1-30 (2006) [http://www.jmlr.org/papers/v7/demsar06a.html]

References

- Bradley Efron (1983), "Estimating the Error Rate of a Prediction Rule: Some Improvements on Cross-Validation," Journal of the American Statistical Association, 78, 316-331.

  DOI: 10.2307/2288636 [http://www.jstor.org/stable/2288636]
- Bradley Efron and Robert Tibshirani “Improvements on Cross-Validation: The .632+ Bootstrap Method” Journal of the American Statistical Association 92 (438), 548-560 (1997)

  DOI: 10.2307/2965703 [http://www.jstor.org/stable/2965703]
- Kohavi, R., 1995. “A study of cross-validation and bootstrap for accuracy estimation and model selection”. International Joint Conference on Artificial Intelligence 2 (12), 11371143.
- Yoshua Bengio and Yves Grandvalet. “No Unbiased Estimator of the Variance of K-Fold Cross-Validation” J. Mach. Learn. Res. 5, 1089-1105 (2004) [http://www.jmlr.org/papers/v5/grandvalet04a.html]
- **Simone Borra, Agostino Di Ciaccio, “Measuring the prediction error. A comparison of cross-validation, bootstrap and covariance penalty methods”, Computational Statistics & Data Analysis 54 (12) 2976-2989 (2010) [https://doi.org/10.1016/j.csda.2010.03.004]**