MTAT.03.227 Machine Learning

Performance evaluation

Sven Laur
University of Tartu

Short list of goodness measures

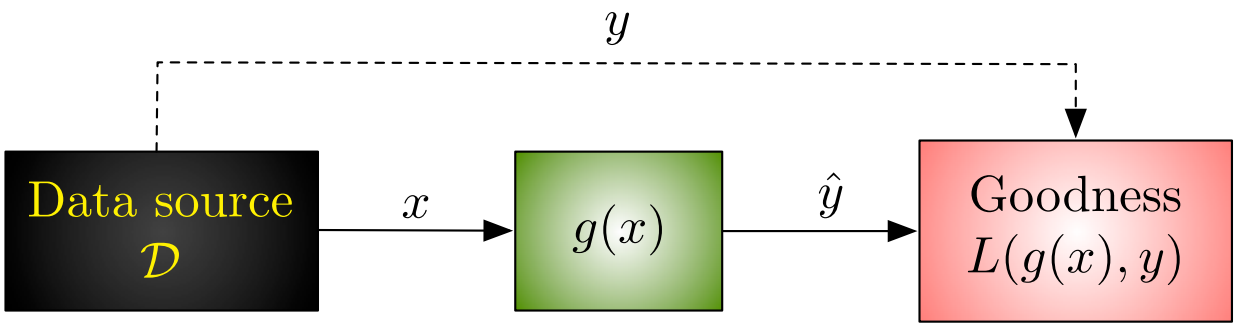
Some goodness measures for classification

- ▷ Accuracy – the percentage of correctly classified examples
- ▷ Precision – the percentage of correct labels among positive guesses
- ▷ Recall – the percentage of positive cases that are detected

Some goodness measures for regression

- ▷ Normalised mean square error
- ▷ Normalised mean absolute error
- ▷ Trimmed mean square and absolute error estimates

How to estimate performance in the future



For any prediction algorithm we can its expected goodness in the future

Practice. Average goodness over a long enough series of future samples.

- ▷ Sampling should not change the data source in the future.
- ▷ All future samples should be independent from each other.

Theory. We should expected goodness over the data distribution

- ▷ The distributions always exists although we might know it.
- ▷ Expected value exists even if the number of future samples is limited.

Theoretical formulation

Let $\mathcal{D}$ be the distribution of (x, y) pairs where x is the input and y is the target of a prediction algorithm g.

Let $L : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ be the *loss function* which takes in the predicted value $\hat{y}$ and the actual value y and outputs resulting loss.

Then the corresponding *risk* $R(g)$ is computed as *mathematical expectation*

$$R(g) := \mathbf{E}_{\mathcal{D}}(L(g(x), y)) = \int_{(x,y)\in\mathcal{D}} L(g(x), y) dF(x, y)$$

where F is the corresponding probability measure.

Practical example

- Let $g(x_1, x_2) \equiv 0$ and let $L(\hat{y}, y) = (y - \hat{y})^2$. What is the risk $R(g)$ if the next data sample is chosen uniformly from the following table.

x_1	x_2	y
0	0	0
0	0	1
0	1	1
1	0	0
1	1	1
0	0	0

- Propose a new prediction rule g_* that minimises the risk.
- Is there always a prediction rule that minimises the risk?

Empirical risk estimation

When the sample $D_N = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ is *representative* then we can approximate risk $R(g)$ with *empirical risk*:

$$R_N(g) = \frac{1}{N} \cdot \sum_{i=1}^{N} L(g(x_i), y_i) \ .$$

Iid sampling assumption. The following conditions assure that the sample data D_N is representative (with high probability).

- ▷ All samples are independent from each other.
- ▷ All samples are drawn from same distribution.
- ▷ Future samples come from the same distributions as the data D_N.

Law of large numbers

Central limit theorem. Let $z_1, \ldots, z_N$ be independent and identical samples form a *real-valued distribution* with a *finite standard deviation* σ and *mean* μ. Then the random variable

$$S = \sqrt{N}\left(\frac{1}{N} \cdot \sum_{i=1}^{N} z_i - \mu\right)$$

converges *in distribution* to normal distribution $\mathcal{N}(mean = 0, sd = \sigma)$.

Translation

Under mild assumptions the empirical risk $R_N(g)$ converges to risk $R(g)$ and we can actually use normal distribution to estimate probabilities:

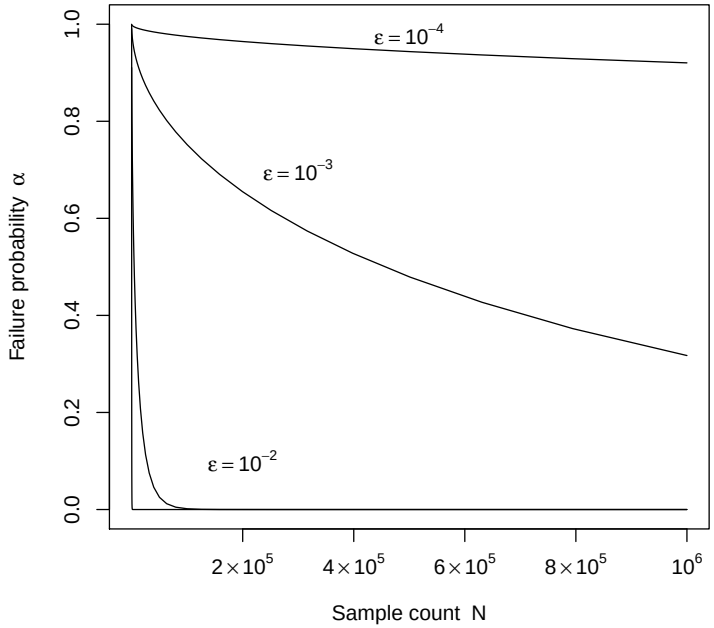
$$\Pr\left[|R_N(g) - R(g)| \geq \varepsilon\right] \lesssim 2 \cdot \int_{-\infty}^{\varepsilon} \frac{\sqrt{N}}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\sqrt{N}t^2}{\sigma^2}\right) dt$$

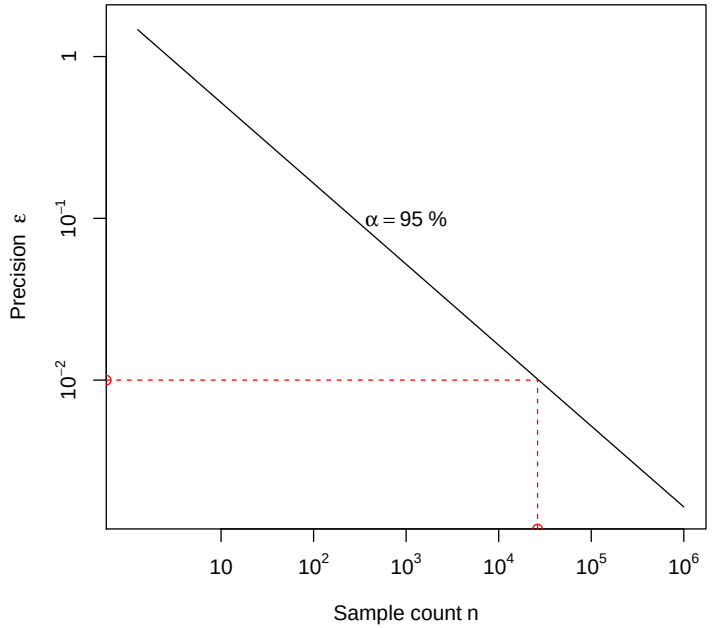
for a finite value σ which is the variance of loss $\mathbf{D}(R(g))$.

Reasoning

- ▷ If (x_i, y_i) are iid samples then $z_i = L(g(x_i), y_i)$ are also iid samples.
- ▷ By definition $\mu = \mathbf{E}(z) = \mathbf{E}(L(g(x), y)) = R(g)$.
- ▷ CLT assumes that risk μ is finite and standard deviation σ is finite.

What does the convergence speed mean





The number of samples needed to get a precision ε is $\mathrm{O}(1/\varepsilon^2)$.

▷ To increase precision 10 times you need 100 times more samples!

Why do we need a test set at all

Machine learning algorithm

- ▷ Count number of zeroes n_0 and number of ones n_1 in training sample.
- ▷ If $n_0 > n_1$ output $g_0(x) \equiv 0$, otherwise output $g_1(x) \equiv 1$.

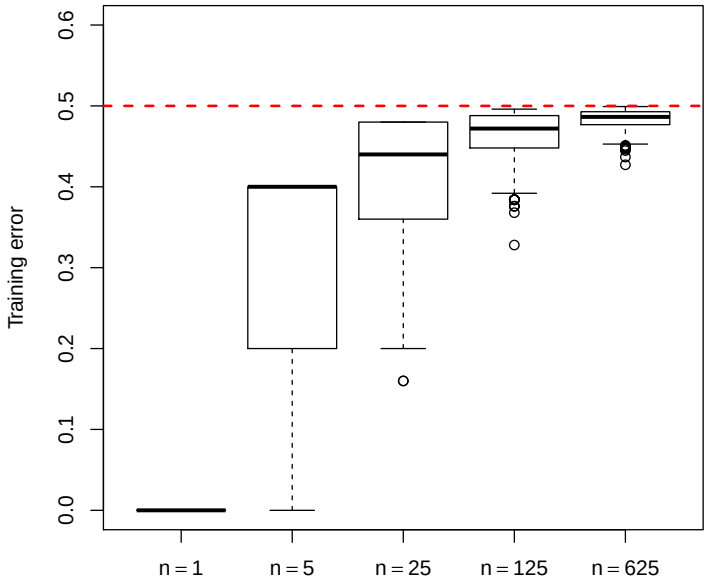
Data source

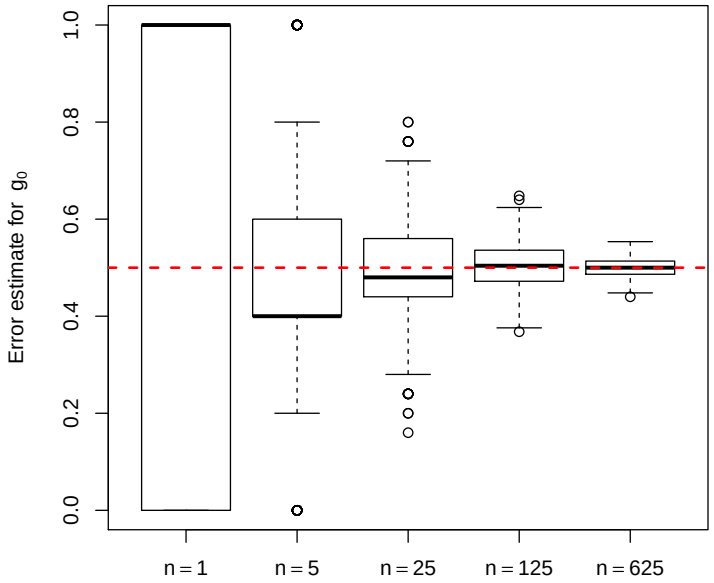
- ▷ Choose the input x randomly form the range $[0, 1]$
- ▷ Choose the label y randomly from the set $\{0, 1\}$.

True risk value

- ▷ Clearly the risk of both rules $R(g_0) = R(g_1) = 0.5$
- ▷ The risk of of our learning algorithm $R(g)$ is also 0.5.

Simulation outcomes for training error

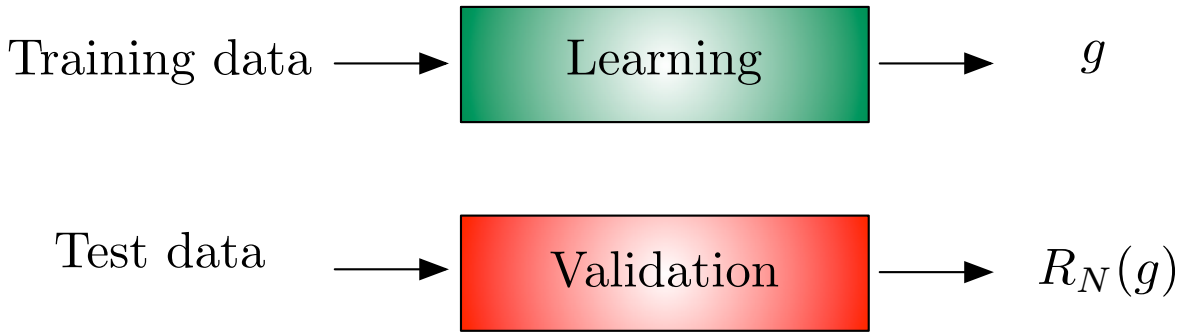




Training error of the rule g is significantly smaller than 0.5

▷ We bias the estimate by choosing the rule g_i for which $R_N(g_i) < R(g)$

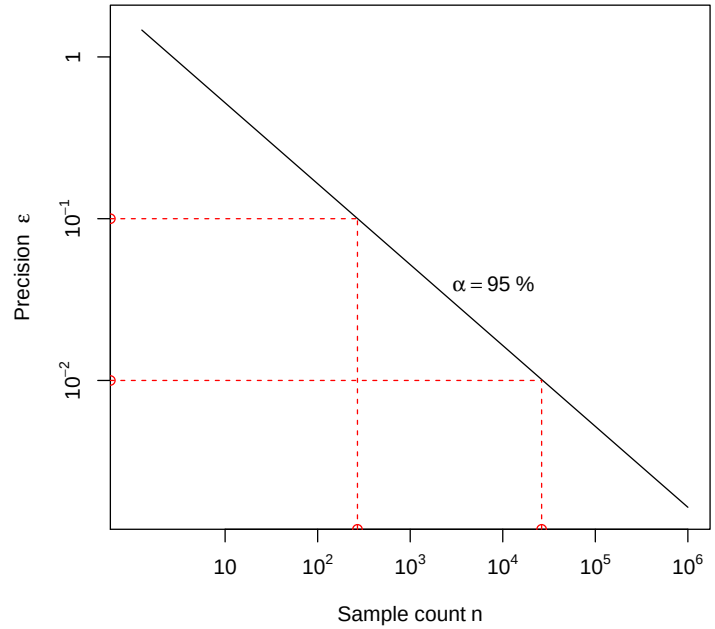
Why does the holdout testing work



By randomly splitting the data into training and test data we assure

- ▷ The training and test sets are independent under iid assumption.
- ▷ On a training set we compare many model and choose few winners.
- ▷ These functions are independent from the test set data.
- ▷ As there number of functions is small the law of large numbers holds.

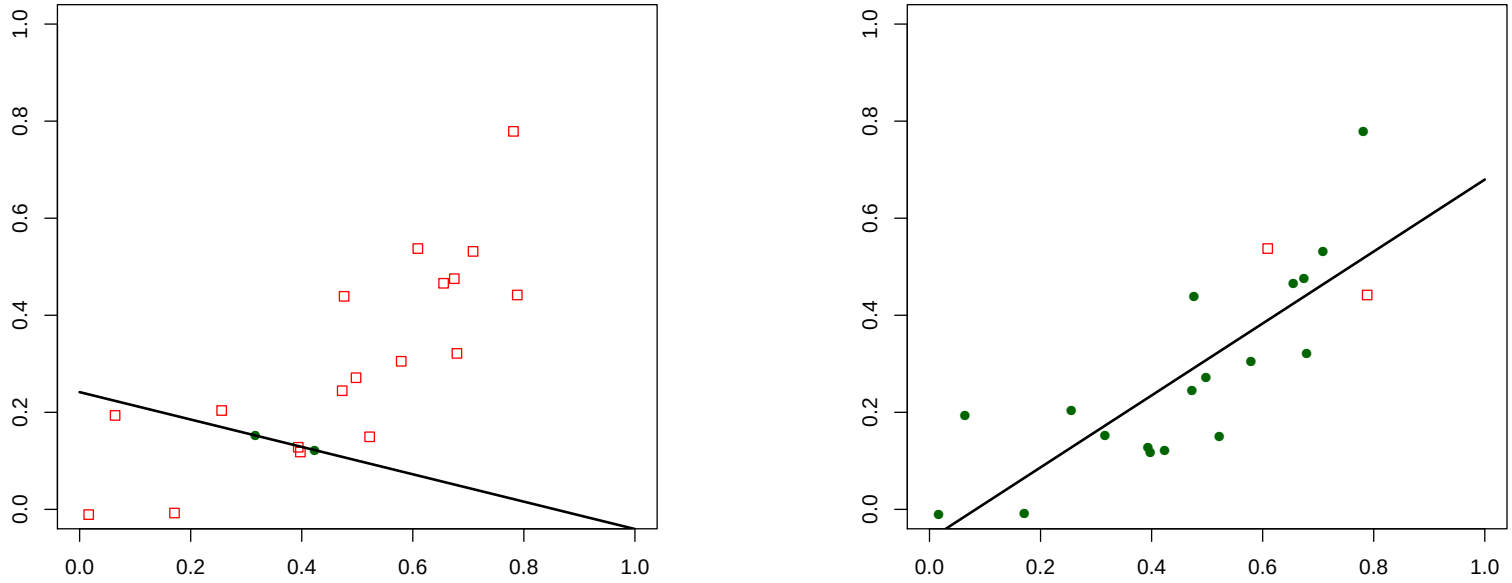
What is the right size of the holdout sample



The holdout sample must be quite large or otherwise the precision is low

- ▷ Roughly 400 data points to get precision 0.1 in classification accuracy.

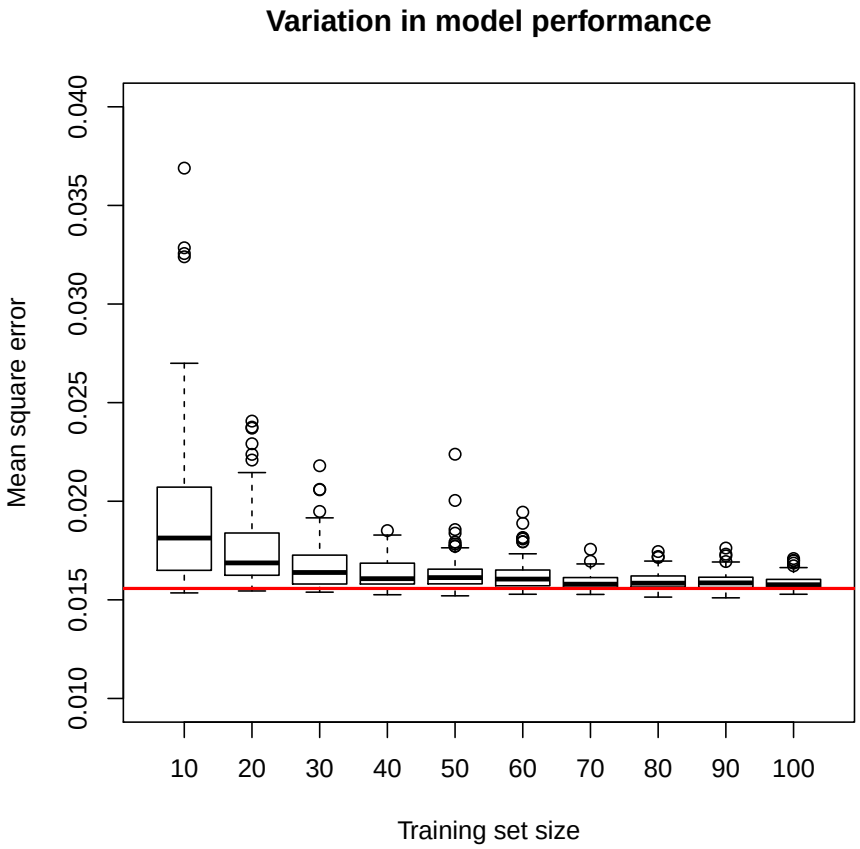
Why is holdout testing problematic

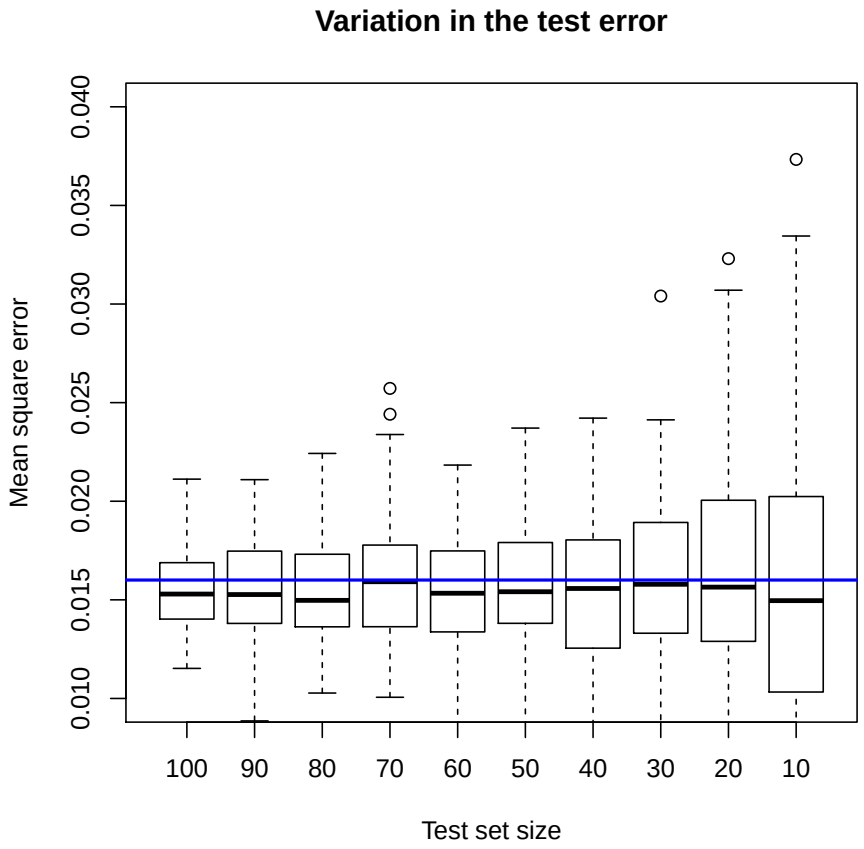


If the number of available data points is small we have to choose:

- ▷ a small training set and bad model but good estimate on risk
- ▷ a big training set and good model but bad estimate on risk

Why is holdout testing problematic





Typical tradeoffs between leaning-bias and variance of the validation error.

Crossvalidation as an engineering trick

To reduce holdout error, we can do several holdout experiments. Since we have not enough data, we redo splitting and training on the same data.

This idea yields a generic crossvalidation scheme

1. Generate several splits of test and training data
2. For each split train the model and compute holdout error
3. Tabulate results

	Split 1	Split 2	...	Split k
Training error	S_1	S_2	...	S_k
Test error	E_1	E_2	...	E_k
Optimism Δ	$E_1 - S_1$	$E_2 - S_2$	...	$E_k - S_k$

4. Compute averages $E = \frac{1}{k}(E_1 + \cdots + E_k)$ and $\Delta = \frac{1}{k}(\Delta_1 + \cdots + \Delta_k)$
5. Visualise results and compute confidence intervals for estimates if needed.

Different flavours of cross validation

Exhaustive data splitting

- ▷ Leave-one-out method, leave-p-out method

Partial splitting

- ▷ K-fold cross validation for $K = 5, 10$
- ▷ Monte-Carlo crossvalidation with a fixed split ratio, e.g $1 : 9$. Same split can occur more than once
- ▷ Repeated learning testing with a fixed split ratio, e.g $1 : 9$. Same split can occur only once.

Different flavours of cross validation

Direct estimation of model error

- ▷ The model error is estimated as an average of test errors E_i.
- ▷ If needed one computes standard deviation estimates of the average error.
- ▷ If needed one draws a boxplot of all test errors.

Model error through the optimism

- ▷ First the optimism Δ is estimated as an average $E_i - S_i$.
- ▷ Next model is trained on the entire data and training error S is recorded.
- ▷ Model error is estimated as $S + \Delta$
- ▷ If needed the standard deviation is computed for the optimism.
- ▷ If needed on draws a shifted boxplot $S + E_i - S_i$ for the error estimate.

Bootstrapping as an alternative

We could use the entire date set for validation if we could get another dataset for training the model. Bootstrapping is an engineering trick to create a new dataset out of a thin air.

1. Draw N samples from the original dataset with replacement to get a *bootstrap sample* D_B, e.g. the same element can occur more than once.
2. Train the model on the bootstrap sample D_B
3. Estimate the test error on the original dataset D
4. Repeate the procedure 20-200 times
5. Compute necessary statistics and visualise the results if needed

Standard way how to use bootstrapping

Bootstrapping is mostly used to estimate optimism

- ▷ The model is trained and the training error S_i is computed
- ▷ The test error E_i is usually computed on the entire dataset
- ▷ Optimism is computed as $E_i - S_i$

Note that it does not make sense to compute test error on the entire dataset as the we have used some of the data to build a model. Advanced bootstrap methods like **.632 bootstrap** and **.632 bootstrap+** use only the out of training set error and later find a tradeoff between training an test error.

$$E_{\text{boot}} = 0.368 \cdot S_{\text{Train}} + 0.632 \cdot E_{\text{Out-of-training-set}}$$

Other uses of bootstrapping

Estimate the noise-tolerance of the machine-learning method

- ▷ Generate a bootstrap sample
- ▷ Corrupt with an appropriate noise
- ▷ Train the model and estimate the performance

Estimate the variance of model coefficients

- ▷ Generate a bootstrap sample
- ▷ Estimate model parameters
- ▷ Visualise parameters and compute empirical quantiles
- ▷ Drop parameter which fluctuate around zero