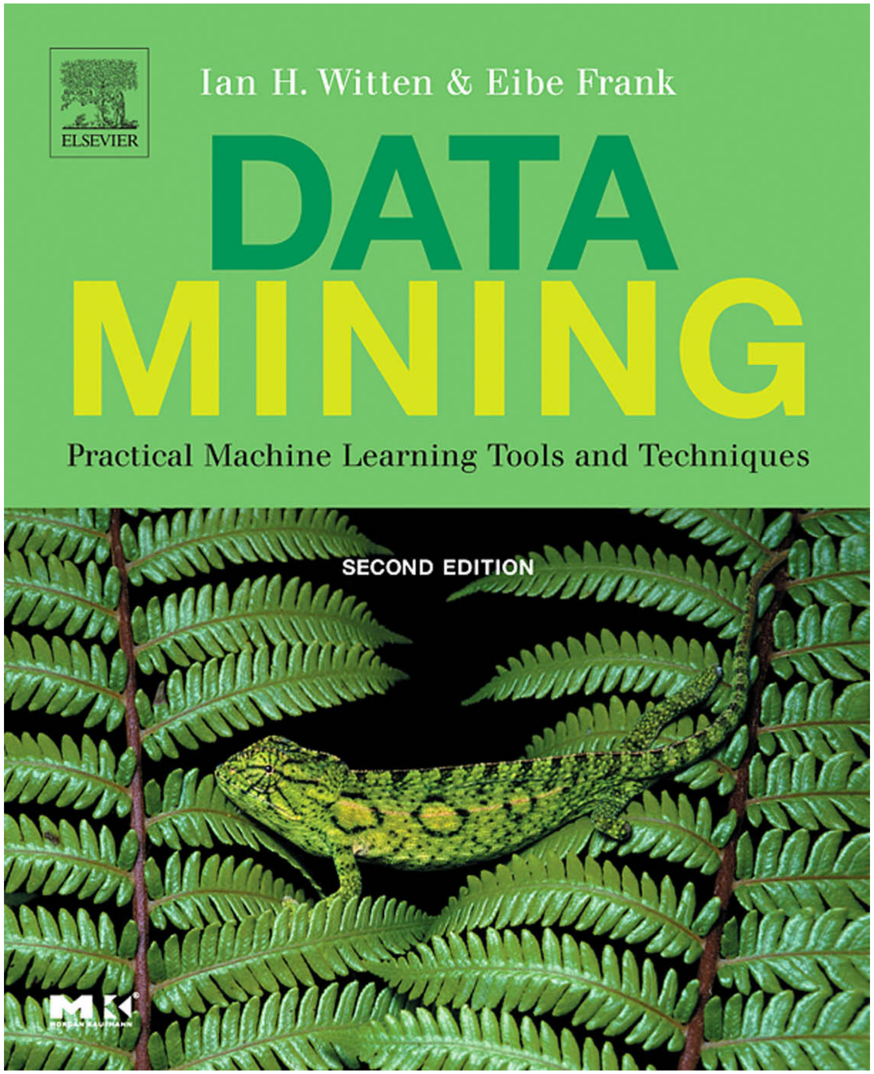


Chapter 5:

Credibility:
Evaluating what’s been learned

Credibility: Evaluating what's been learned

- Training and testing
- Predicting performance
- Cross validation
- Other estimates: Leave-one-out & The bootstrap
- Comparing data mining methods
- Predicting probabilities: loss functions
- Costsensitive measures
- Evaluating numeric prediction
- The Minimum Description Length principle

Evaluation: the key to success

- Error on the training data is *not* a good indicator of performance on future data
  - Otherwise 1NN would be the optimum classifier!
- Simple solution that can be used if lots of data is available:
  - Split data into training and test set
- However: data is usually limited
  - More sophisticated techniques need to be used

Issues in evaluation

- Statistical reliability of estimated differences in performance (-> significance tests)
- Choice of performance measure:
  - Number of correct classifications
  - Accuracy of probability estimates
  - Error in numeric predictions
- Costs assigned to different types of errors
  - Many practical applications involve costs

5.1 Training and testing

Training and testing

- Natural performance measure for classification problems: *error rate*
  - *Success*: instance's class is predicted correctly
  - *Error*: instance's class is predicted incorrectly
  - Error rate: proportion of errors made over the whole set of instances
- *Resubstitution error:* error rate obtained from training data
- Resubstitution error is optimistic!

Training and testing

- *Test set*: independent instances that have played no part in formation of classifier
  - Assumption: both training data and test data are representative samples of the underlying problem
- Test and training data may differ in nature
  - Example: classifiers built using customer data from two different towns *A* and *B*
    - To estimate performance of classifier from town *A* in completely **new town**, test it on data from *B*

Note on parameter tuning

- It is important that the test data is not used *in any way* to create the classifier
- Some learning schemes operate in two stages:
  - Stage 1: build the basic structure
  - Stage 2: optimize parameter settings
- The test data can't be used for parameter tuning!
- Proper procedure uses *three* sets: *training data*, *validation data*, and *test data*
  - Training data is used to build the basic structure
  - Validation data is used to optimize parameters or to select a particular method
  - Test data is used to calculate the error rate of the final method

Making the most of the data

- Once evaluation is complete, *all the data* can be used to build the final classifier
- Generally,
  - The larger the training data the better the classifier
  - The larger the test data the more accurate the error estimate
- *Holdout* procedure: method of splitting original data into training and test set
  - Dilemma: ideally both training set *and* test set should be large!

5.2 Predicting performance

Predicting performance

- Assume the estimated error rate is 25%. How close is this to the true error rate?
  - Depends on the amount of test data
- Prediction is just like tossing a (biased) coin "Head" is a "success", "tail" is an "error"
- In statistics, a succession of independent events like this is called a *Bernoulli process*
  - Statistical theory provides us with confidence intervals for the true underlying proportion

Confidence intervals

- Suppose p is success rate, that out of N trials, S are successes: thus the observed success rate is $f = S/N$
- We can say: p lies within a certain specified interval with a certain specified confidence
- Example: S=750 successes in N=1000 trials
  - Estimated success rate: 75%
  - How close is this to true success rate p?
    - Answer: with 80% confidence p in [73.2,76.7]
- Another example: S=75 and N=100
  - Estimated success rate: 75%
  - With 80% confidence p in [69.1,80.1]

Mean and variance

- Mean and variance for a Bernoulli trial: p, $p(1-p)$
- Expected success rate $f=S/N$
- Mean and variance for f : p, $p(1-p)/N$
- For large enough N, f follows a Normal distribution
- c% confidence interval $[-z \le X \le z]$ for random variable with 0 mean is given by:

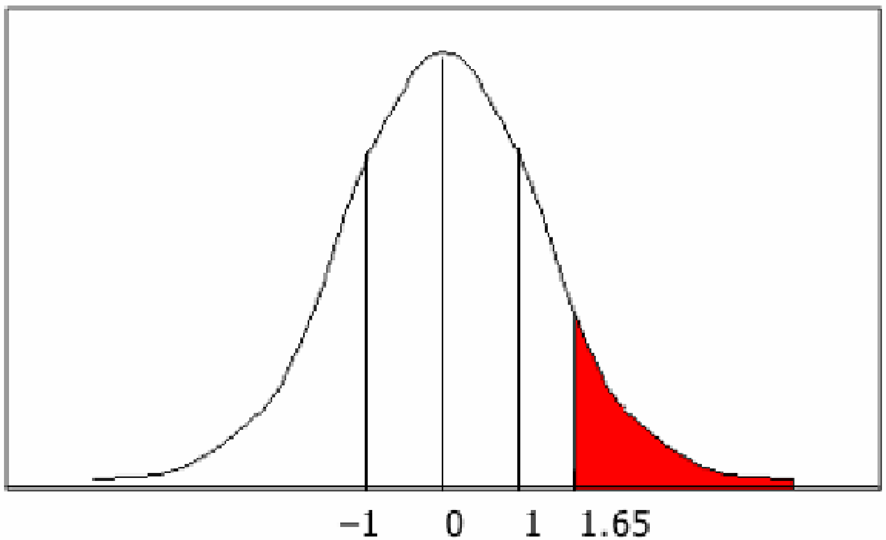
$$\Pr[-z \le X \le z] = c$$

- With a symmetric distribution:

$$\Pr[-z \le X \le z] = 1 - 2 \times \Pr[x \ge z]$$

Confidence limits

- Confidence limits for the normal distribution with 0 mean and a variance of 1:



$\Pr[X \geq z]$	z
0.1%	3.09
0.5%	2.58
1%	2.33
5%	1.65
10%	1.28
20%	0.84
40%	0.25

- Thus: $\Pr[-1.65 \leq X \leq +1.65] = 90\%$
- To use this we have to reduce our random variable *f* to have 0 mean and unit variance

Transforming *f*

- Transformed value for *f* to have zero mean and unit variance

$$\frac{f-p}{\sqrt{p(1-p)/N}}$$

  – Subtract the mean and divide by the *standard deviation*

- Resulting equation:

$$\Pr\left[-z<\frac{f-p}{\sqrt{p(1-p)/N}}<z\right]=c$$

- Solving for *p* :

$$p=\left(f+\frac{z^2}{2N}\pm z\sqrt{\frac{f}{N}-\frac{f^2}{N}+\frac{z^2}{4N^2}}\right)\Bigg/\left(1+\frac{z^2}{N}\right)$$

Examples

- f = 75%, N = 1000, c = 80% (so that z = 1.28):
  Interval for p [0.732, 0.767]
- f = 75%, N = 100, c = 80% (so that z = 1.28):
  Interval for p [0.691,0.801]
- f = 75%, N = 10, c = 80% (so that z = 1.28):
  Interval for p [0.549,0.881]

- Note that normal distribution assumption is only valid for large N (i.e. N > 100)

5.3 Cross-validation

Holdout estimation

- What to do if the amount of data is limited?
- The *holdout* method reserves a certain amount for testing and uses the remainder for training
  - Usually: one third for testing, the rest for training
- Problem: the samples might not be representative
  - Example: class might be missing in the test data
- Advanced version uses *stratification*
  - Ensures that each class is represented with approximately equal proportions in both subsets

Repeated holdout method

- Holdout estimate can be made more reliable by repeating the process with different subsamples
  - In each iteration, a certain proportion is randomly selected for training
  - The error rates on the different iterations are averaged to yield an overall error rate
- This is called the *repeated holdout* method
- Still not optimum: the different test sets overlap
  - Can we prevent overlapping?

Cross-validation

- *Cross-validation* avoids overlapping test sets
  - First step: split data into *k* subsets of equal size
  - Second step: use each subset in turn for testing, the remainder for training
- Called *k-fold cross-validation*
- Often the subsets are stratified before the cross-validation is performed
- The error estimates are averaged to yield an overall error estimate

More on cross-validation

- Standard method for evaluation: stratified ten-fold cross-validation
- Why ten?
  - Extensive experiments have shown that this is the best choice to get an accurate estimate
  - There is also some theoretical evidence for this
- Stratification reduces the estimate's variance
- Even better: repeated stratified cross-validation
  - E.g. ten-fold cross-validation is repeated ten times and results are averaged (reduces the variance)

5.4 Other estimates

Leave-One-Out cross-validation

- Leave-One-Out: a particular form of cross-validation:
  - Set number of folds to number of training instances
  - I.e., for *n* training instances, build classifier *n* times
- Advantages:
  - Makes best use of the data for training in each case
  - Involves no random subsampling

Leave-One-Out-CV and stratification

- Disadvantage of Leave-One-Out-CV:
  - Very computationally expensive
  - It *guarantees* a non-stratified sample because there is only one instance in the test set!
- Extreme example: random dataset split equally into two classes
  - Best inducer predicts majority class
  - 50% accuracy on fresh data
  - Leave-One-Out-CV estimate is 100% error!

The bootstrap

- CV uses sampling *without replacement*
  - The same instance, once selected, can not be selected again for a particular training/test set
- The *bootstrap* uses sampling *with replacement* to form the training set
  - Sample a dataset of *n* instances *n* times *with replacement* to form a new dataset of *n* instances
  - Use this data as the training set
  - Use the instances from the original dataset that don't occur in the new training set for testing

The 0.632 bootstrap

- The *0.632 bootstrap*
  - A particular instance has a probability of 1–1/*n* of *not* being picked
  - Thus its probability of ending up in the test data is:

$$\left(1-\frac{1}{n}\right)^n \approx e^{-1} = 0.368$$

  - Where *e* is the base of natural logarithms, 2.7183
  - This means the training data will contain approximately 63.2% of the instances

Estimating error with the bootstrap

- The error estimate on the test data will be very pessimistic
  - Trained on just ~63% of the instances
- Therefore, combine it with the resubstitution error:

$$e = 0.632 \times e_{\text{test instances}} + 0.368 \times e_{\text{training instances}}$$

- The resubstitution error gets less weight than the error on the test data
- Repeat process several times with different replacement samples; average the results

More on the bootstrap

- Probably the best way of estimating performance for very small datasets
- However, it has some problems
  - Consider the random dataset from above
  - A perfect memorizer will achieve
    0% resubstitution error and
    ~50% error on test data
  - Bootstrap estimate for this classifier:
    $err = 0.632 \times 50\% + 0.368 \times 0\% = 31.6\%$
  - True expected error: 50%

5.5 Comparing data mining methods

Comparing data mining methods

- Frequent question: which of two learning methods performs better?
- Note: this is domain dependent!
- Obvious way: compare 10-fold CV estimates
- Generally sufficient in applications
- How about, when a new learning algorithm is proposed?
  - Need to show that a particular method works really better

Comparing data mining methods (II)

- Want to show that method A is better than method B in a particular domain
  - For a given amount of training data
  - On average, across all possible training sets
- Let's assume we have an infinite amount of data from the domain:
  - Sample infinitely many dataset of specified size
  - Obtain cross-validation estimate on each dataset for each method
  - Check if mean accuracy for method A is better than mean accuracy for method B

Paired t-test

- In practice we have limited data and a limited number of estimates for computing the mean
- *Student's t-test* tells whether the means of two samples are significantly different
- In our case the samples are cross-validation estimates for different datasets from the domain
- Use a *paired* t-test because the individual samples are paired
  - The same CV is applied twice

5.6 Predicting probabilities

Predicting probabilities

- Performance measure so far: success rate
- Also called *0-1 loss function,* the “loss” is:
  - 0 if prediction is correct
  - 1 if prediction is incorrect
- Some classifiers produces class probabilities (such as the Naïve Bayes method)
- Depending on the application, we might want to check the accuracy of the probability estimates
- 0-1 loss is not the right thing to use in those cases

Quadratic loss function

- Suppose that for a single instance there are *k* possible classes
- $p_1 \ldots p_k$ are probability estimates for an instance classes
- *c* is the index of the instance's actual class
- $a_1 \ldots a_k = 0$, except for a_c which is 1
- *Quadratic loss* is:

$$\sum_j (p_j - a_j)^2$$

5.7 Counting the cost

Counting the cost

- In practice, different types of classification errors often incur different costs
- Examples:
  - Terrorist profiling
    - "Not a terrorist" correct 99.99% of the time
  - Loan decisions
  - Oil-slick detection
  - Fault diagnosis
  - Promotional mailing

Counting the cost

- The *confusion matrix*:

		Predicted class	
		yes	no
Actual class	yes	true positive	false negative
	no	false positive	true negative

- The overall success rate is:

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

Classification with costs

- Default cost matrixes for the two- and three-class cases

		Predicted class					Predicted class	
		yes	no			a	b	c
Actual class	yes	0	1	Actual class	a	0	1	1
	no	1	0		b	1	0	1
					c	1	1	0

- Success rate is replaced by average cost per prediction
  - Cost is given by appropriate entry in the cost matrix

Cost-sensitive classification

- Can take costs into account when making predictions
  - Basic idea: only predict high-cost class when very confident about prediction
- Given: predicted class probabilities
  - Normally we just predict the most likely class
  - Here, we should make the prediction that minimizes the expected cost
    - Expected cost: dot product of vector of class probabilities and appropriate column in cost matrix
    - Choose column (class) that minimizes expected cost

Cost-sensitive learning

- So far we haven't taken costs into account at training time
- Most learning schemes do not perform cost-sensitive learning
  - They generate the same classifier no matter what costs are assigned to the different classes
  - Example: standard decision tree learner
- Simple methods for cost-sensitive learning:
  - Resampling of instances according to costs
  - Weighting of instances according to costs

Lift charts

- In practice, costs are rarely known
- Decisions are usually made by comparing possible scenarios
- Example: promotional mailout to 1,000,000 households
  - Mail to all; 0.1% respond (1000)
  - Data mining tool identifies subset of 100,000 most promising, 0.4% of these respond (400)
    40% of responses for 10% of cost may pay off
  - Identify subset of 400,000 most promising, 0.2% respond (800)
- A *lift chart* allows a visual comparison

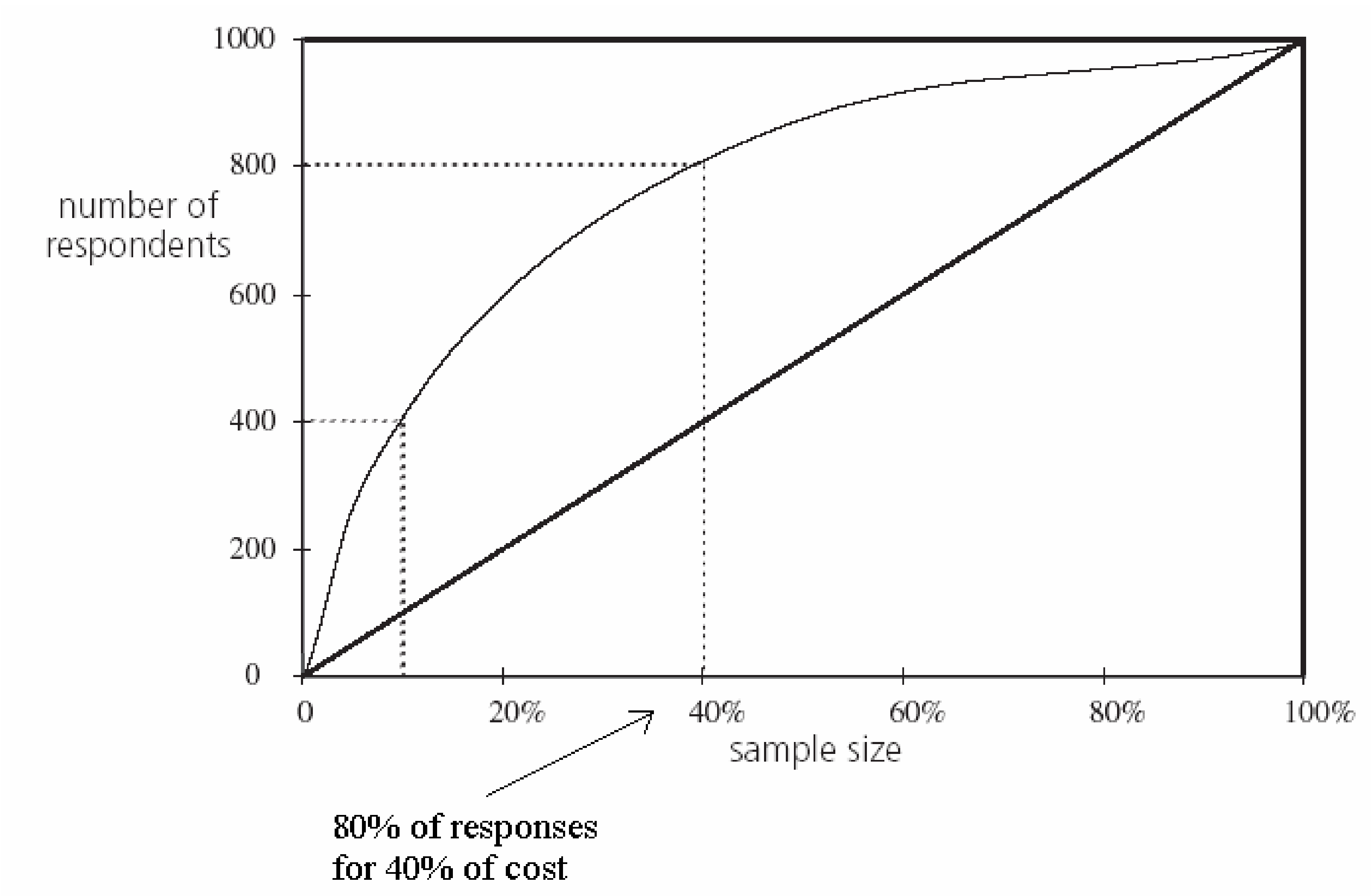
Generating a lift chart

- Sort instances according to predicted probability of being positive:

Rank	Predicted probability	Actual class	Rank	Predicted probability	Actual class
1	0.95	*yes*	11	0.77	*no*
2	0.93	*yes*	12	0.76	*yes*
3	0.93	*no*	13	0.73	*yes*
4	0.88	*yes*	14	0.65	*no*
5	0.86	*yes*	15	0.63	*yes*
6	0.85	*yes*	16	0.58	*no*
7	0.82	*yes*	17	0.56	*yes*
8	0.80	*yes*	18	0.49	*no*
9	0.80	*no*	19	0.48	*yes*
10	0.79	*yes*	...	...	...

- A small dataset with 150 instances, of which 50 are *yes* responses—an overall success proportion of 33%.

A hypothetical lift chart



5.8 Evaluating numeric prediction

Evaluating numeric prediction

- Strategies: independent test set, cross-validation, significance tests, etc.
- Difference: error measures
- Actual target values: $a_1\ a_2 \ldots a_n$
- Predicted target values: $p_1\ p_2 \ldots\ p_n$
- Most popular measure: *mean-squared error*

$$\frac{(p_1 - a_1)^2 + \ldots + (p_n - a_n)^2}{n}$$

  – Easy to manipulate mathematically

Other measures

- The *root mean-squared error* :

$$\sqrt{\frac{(p_1-a_1)^2+\ldots+(p_n-a_n)^2}{n}}$$

- The *mean absolute error:*

$$\frac{|p_1-a_1|+\ldots+|p_n-a_n|}{n}$$

  – is less sensitive to outliers than the mean-squared error:

Improvement on the mean

- How much does the scheme improve on simply predicting the average?

- The *relative squared error* is:

$$\frac{(p_1 - a_1)^2 + \ldots + (p_n - a_n)^2}{(a_1 - \bar{a})^2 + \ldots + (a_n - \bar{a})^2}, \text{ where } \bar{a} = \frac{1}{n}\sum_i a_i$$

- The *relative absolute error* is:

$$\frac{|p_1 - a_1| + \ldots + |p_n - a_n|}{|a_1 - \bar{a}| + \ldots + |a_n - \bar{a}|}$$

Correlation coefficient

- Measures the *statistical correlation* between the predicted values and the actual values

$$\frac{S_{PA}}{\sqrt{S_P S_A}}, \text{ where } S_{PA} = \frac{\sum_i (p_i - \overline{p})(a_i - \overline{a})}{n-1},$$

$$S_p = \frac{\sum_i (p_i - \overline{p})^2}{n-1}, \text{ and } S_A = \frac{\sum_i (a_i - \overline{a})^2}{n-1}$$

- Scale independent, between –1 and +1
- Good performance leads to large values!

Which measure?

- Best to look at all of them
- Often it doesn't matter
- Example: Performance measures for four numeric prediction models

	A	B	C	D
root mean-squared error	67.8	91.7	63.3	57.4
mean absolute error	41.3	38.5	33.4	29.2
root relative squared error	42.2%	57.2%	39.4%	35.8%
relative absolute error	43.1%	40.1%	34.8%	30.4%
correlation coefficient	0.88	0.88	0.89	0.91

The end of

Chapter 5: Credibility: Evaluating what’s been learned