DM & ML
Credibility: Evaluating what's been learned

Gergely Lukács

Pázmány Péter Catholic University

Faculty of Information Technology and Bionics

Budapest, Hungary

lukacs@itk.ppke.hu

Evaluation: the key to success

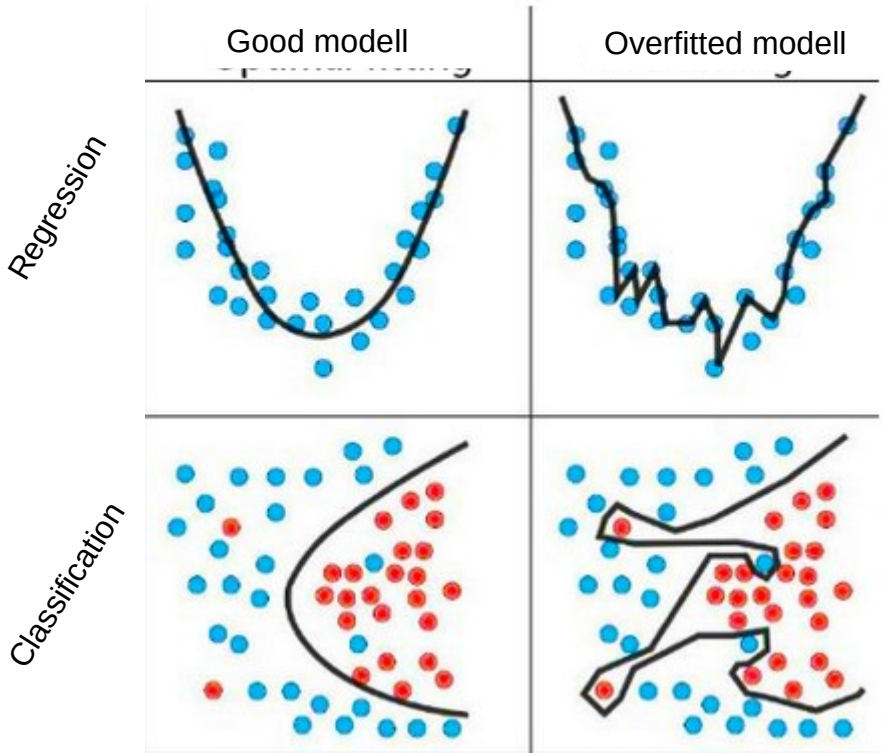
- How predictive is the model we learned?

Also: Basis for developing useful data mining algorithms

Basic performance measure

- Natural performance measure for classification: *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

Overfitting



- Model fits training data well, but only training data; but: model generalizes poorly, consequently, bad prediction for data not used for model building
- **model too complex** and has a **bad prediction performance**

*„Overfitting is when you have a **complicated model** that gives **worse predictions**, on average, than a simpler model" (Andrew Gelman)*

Overfitting - resubstitution error

- Error on the training data – ***resubstitution error*** – is *not* a good indicator of performance on future data

- Resubstitution error is (hopelessly) optimistic!

Test Set

- *Test set*: set of 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
- It is important that the test data is not used *in any way* to create the classifier used for performance evaluation („data leakage”)

Test Set in Practice

- 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 a completely new town, test it on data from *B*
- Changes over time

Performance measures: Binary classification

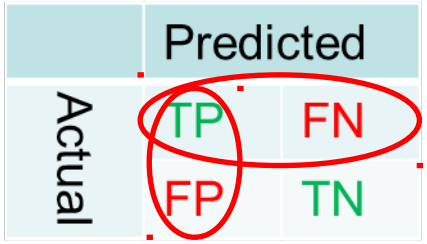
The *confusion matrix*

		Predicted class	
		Yes	No
Actual class	Yes	*True positive (TP)*	*False negative (FN, Type II error)*
	No	*False positive (FP, Type I error)*	*True negative (TN)*

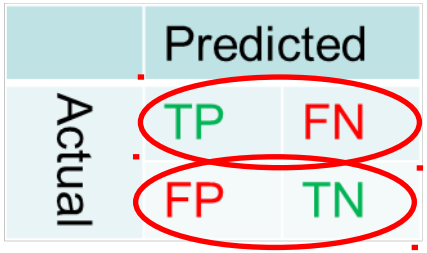
Measures

- *Success rate:* (TP + TN)/(TP+FP+TN+FN)
- *Error rate:* (FP + FN)//(TP+FP+TN+FN)

- Focus on true positives
  used in information retrieval
  (selecting relevant „document”)
  - Proportion of true positives
    to all retrieved
    precision=TP/(TP +FP)
  - Proportion of (retrieved) true positives
    to all positives
    recall = TP /(TP + FN)



- Focus on positives and negatives
  used in medical testing
  - Proportion of positives that are correctly identified:
    Sensitivity (True Positive Rate) = TP / (TP + FN) (==recall)
  - Proportion of negatives that are correctly identified
    Specificity (True Negative Rate) = TN / (FP + TN)



Multiclass problems

- Some measures work directly
  - confusion matrix
  - success rate
  - error rate
- Some measures have to be adapted/extendend
  - Precision values for each of the classes
  - Average precision
  - Precision, recall …

Unbalanced classes?

- Medical test, 1% of cases positive, 99% negative
- If we always predict negative, we have a success rate of 99% !!

Taking costs into account

- Real problem
  - Different types of errors not equally bad
  - Medical test …
  - Aeroplane fault diagnosis (missing an engine failout?)
  - Terrorist profiling (not terrorist correct 99.99% of the time, but missing one has very high costs!)
  - …
- Success rate is replaced by average cost per prediction
  - Cost is given by appropriate entry in the cost matrix

		Predicted class	
		yes	*no*
Actual class	*yes*	0	**5**
	no	1	0

Cost-sensitive 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

Modifying learning for considering costs

- Most learning algorithms generate the same classifier, independently of the cost matrix
  - Cost matrix only influences final prediction

- 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

- Considering cost matrix during training (and ignoring during prediction)
  - Simple methods for cost-sensitive learning:
    - Resampling of instances according to costs
    - Weighting of instances according to costs
  - Some algorithmscan take costs into account by varying a parameter,
    - e.g. naïve Bayes

Probability estimation performance

- Performance measure so far:
  - success rate, i.e. „*0-1 loss function*”:

$$\sum_i \begin{cases} 0 \text{ if prediction is correct} \\ 1 \text{ if prediction is incorrect} \end{cases}$$

- Some classifiers produce **class probabilities**
- Depending on the application, we might want to check the **accuracy of the probability estimates**
  - rather than simply comparing them to a fixed threshold, 0-1 loss
  - e.g. prediction incorrect when class probability ~1 vs ~.51

Probability estimation performance 1: Cross entropy

- average number of bits needed to identify an event drawn from the set, if a coding scheme is used that is optimized for predicted probability distribution, rather than the "true" distribution
- For single instance
  - $-\log(p_c)$,where c is the index of the instance's actual class
- Minimum is reached at true probabilities

Probability estimation performance 2: Mean Quadratic Loss / Brier Score

- For a single instance:

$$E\left[\sum_j (p_j - a_j)^2\right] = \left(\sum_{j \neq c} p_j^2\right) + (1 - p_c)^2$$

  - $p_1, \ldots, p_k$ are probability estimates for an instance
  - c is the index of the instance's actual class
  - $a_1, \ldots, a_k = 0$, except for a_c, which is 1
- For test set: mean quadratic loss or Brier Score

- Can show that this is minimized when $p_j = p_j^*$, the true probabilities

Cross entropy vs Brier score

- probability of single class vs all probabilities
- unbounded (values up to infinity) vs bounded to 2
- Standard: cross entropy
- Argument for Brier:
  - you seek to describe model performance using a measure the model was not optimizing
  - avoiding infinity

Limits of performance predictions

- Example:
  - The estimated error rate is 25%.
    How close is this to the true error rate?
  - *S*=750 successes in *N*=1000 trials
    - Estimated success rate (p): 75%
  - *S*=75 and *N*=100
    - Estimated success rate (p): 75%
  - Depends on the amount of test data
- We can say: *p* lies within a certain specified interval with a certain specified confidence

Bernoulli trial

- Prediction is just like tossing a **biased** coin
  - "Head" is a "success", "tail" is an "error"
  - *Bernoulli trial*, *Bernoulli process*
- For a Bernoulli trial:
  - Mean: p, variance p(1-p)
- Expected success rate f=S/N
  - Mean: p, variance $p\ (1–p)/N$
- For large enough N, f follows a Normal distribution
- c% confidence interval $[–z =< X =< z]$ for random variable with 0 mean is given by:

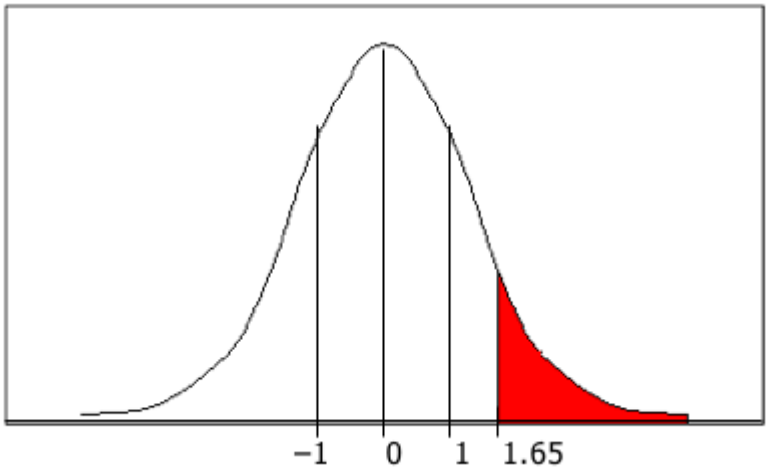
$$Pr[-z \leq X \leq z] = c$$

- Symmetric distribution:

$$Pr[-z \leq X \leq z] = 1 - 2 \times Pr[x \geq z]$$

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

$$Pr[-1.65 \leq X \leq 1.65] = 90\,\%$$

- To use this we have to reduce our random variable *f* to have 0 mean and unit variance

Transforming *f*

- Transformed value for *f* : $\frac{f-p}{\sqrt{p(1-p)/N}}$

  (i.e. subtract the mean and divide by the *standard deviation*)
- Resulting equation:

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

- Solving for *p* :

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

Examples

- f = 75%, N = 1000, c = 80% (so that z = 1.28):

$$p \in [0.732, 0.767]$$

- f = 75%, N = 100, c = 80% (so that z = 1.28):

$$p \in [0.691, 0.801]$$

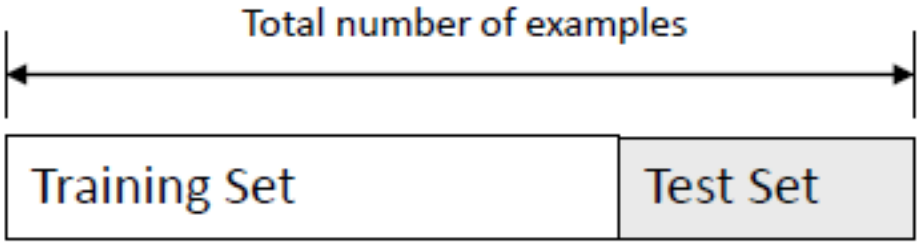
- Note that normal distribution assumption is only valid for large N (i.e. $N > 100$)
- f = 75%, N = 10, c = 80% (so that z = 1.28):

$$p \in [0.549, 0.881]$$

?? (Normal distribution N > 100)

Holdout method

- Method of splitting original data into training and test set
- The *holdout* method reserves a certain amount for testing and uses the remainder for training



Making the most of the data

- Generally, the larger the training data the better the classifier (but with diminishing return)
- The larger the test data the more accurate the error estimate

- Usually: one third for testing, the rest for training

- Once evaluation is complete, *all the data* can be used to build the final classifier (for „simple” classifiers)
  - Performance is improved over the previously estimated level

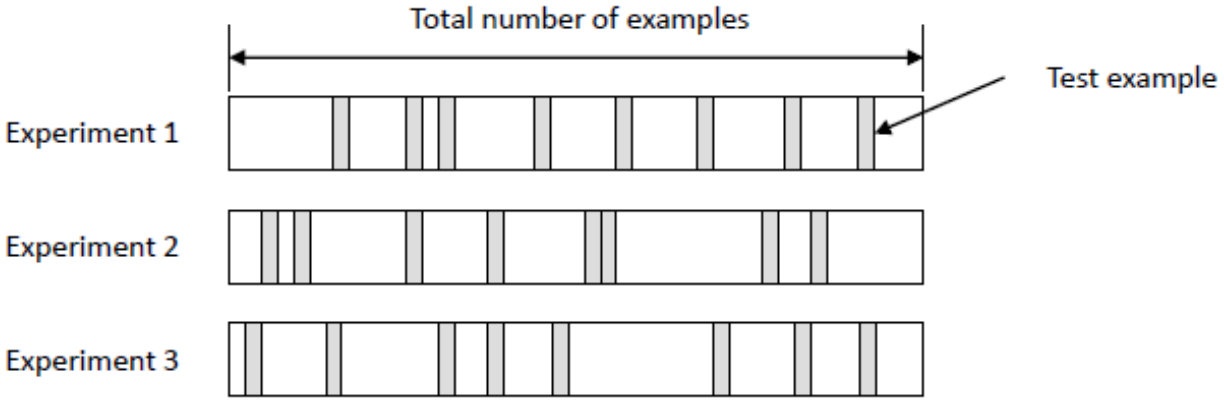
Problems

- The samples might not be representative
  - Example: some class might be missing in the test data
- Modification: ***stratification***
  - Ensures that each class is represented with approximately equal proportions in both subsets

- Data is usually (costly and) limited! (e.g., human/expert knowledge for creating labelled data)
  - More sophisticated techniques need to be used
- Family of resampling methods at the expense of higher computational cost
  - Cross validation
    - (Random subsampling)
    - K-fold cross-validation
    - Leave-one-out cross-validation
  - Bootstrap

Repeated holdout method; Random subsampling

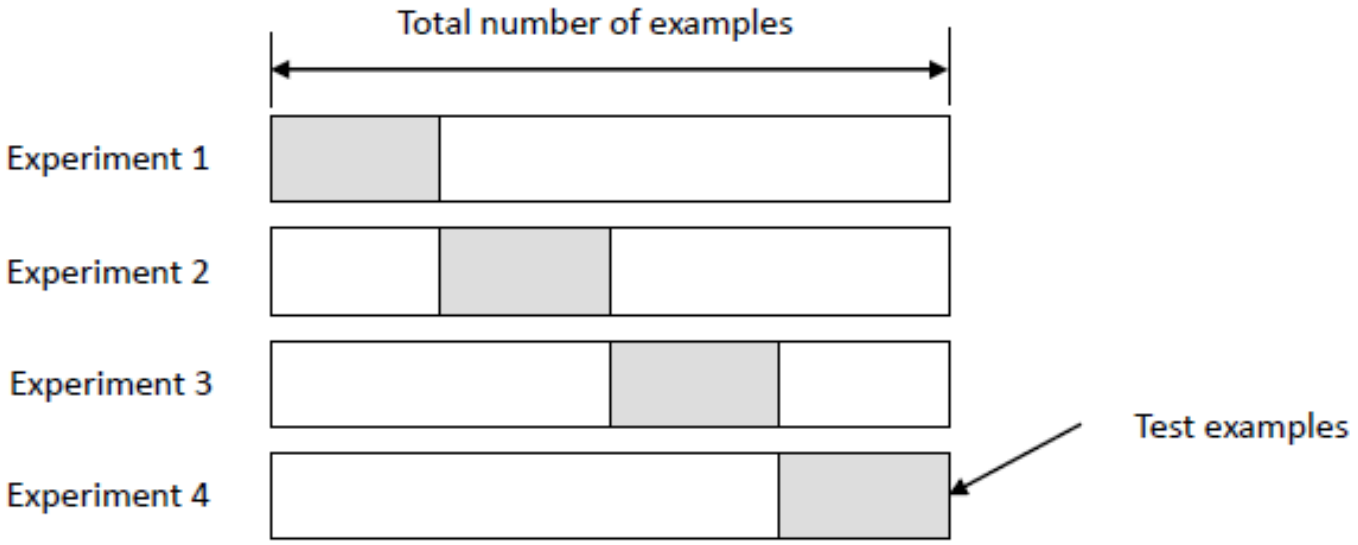
- K data splits of the entire dataset
  - Each data split randomly selects a (fixed) number of examples without replacement
  - For each data split we retrain the classifier from scratch with the training examples and then estimate E_i with the test examples



- The true error estimate is obtained as the average of the separate estimates $E = \frac{1}{K}\sum_{i=1}^{K} E_i$
- This estimate is significantly better than the holdout estimate
- Still not optimum: the different test sets overlap

K-fold Cross Validation

- K-fold cross validation
  1. data is split into *k* subsets of equal size
  2. Second step: each subset in turn is used for testing and the remainder for training



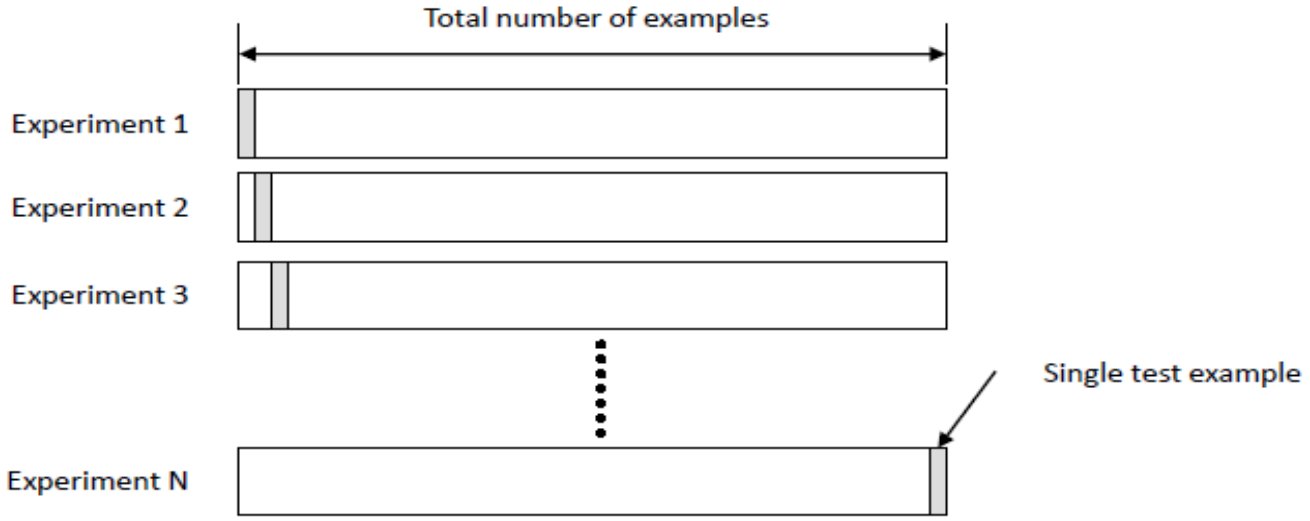
- The error estimates are averaged (as before)
- Often the subsets are stratified before the cross-validation is performed

More on cross-validation

- Standard method for evaluation: stratified ten-fold cross-validation
- Why **ten**-fold? 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) – for performance estimation

Leave-one-out cross-validation (LOO-CV)

- Leave-one-out cross-validation is a degenerate case of K-Fold CV: The number of folds is set to the number of training instances



- Advantages
  - Makes maximum use of the data
  - No random subsampling involved
- Computationally very expensive (in general)
  - for some DM algorithms it can be efficient

LOO-CV and stratification

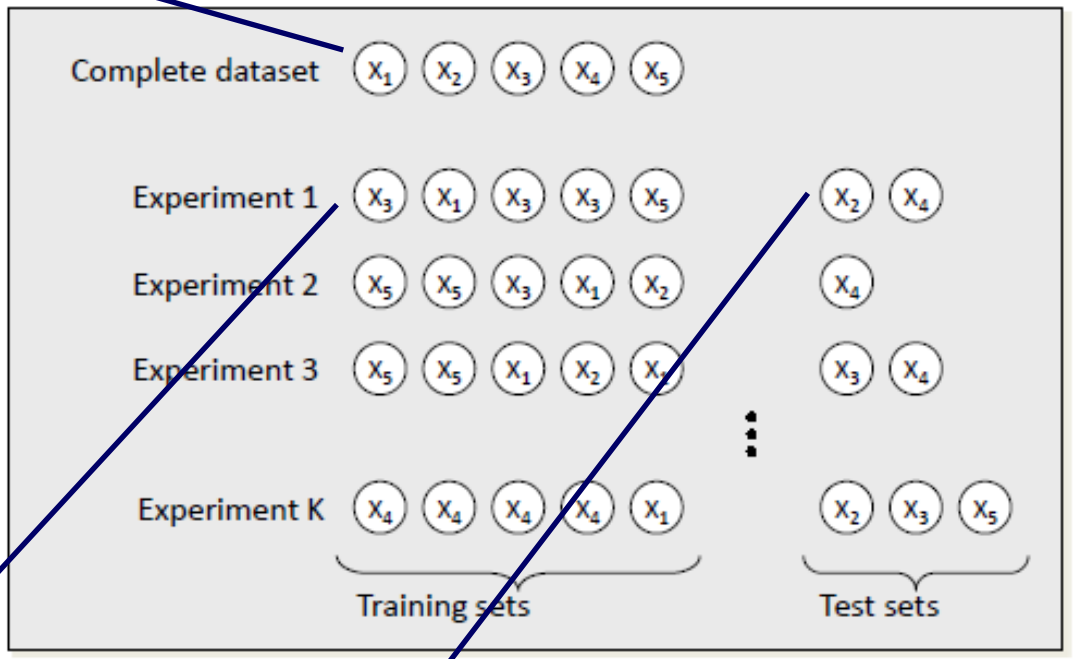
- Disadvantage of LOO-CV: stratification is not possible
  - It *guarantees* a non-stratified sample because there is only one instance in the test set!
- Extreme example: completely random dataset with two classes and equal proportions for both of them
  - Best inducer predicts majority class
  - 50% accuracy on fresh data
  - LOO-CV error rate estimate for this inducer will be 100%!

The bootstrap

- CV uses sampling *without replacement*
- The *bootstrap* is an estimation method that uses sampling with replacement to form the training set
  - A dataset of *n* instances is sampled *n* times with replacement to form a new dataset of *n* instances
  - This data is used as the training set
  - The instances from the original dataset that don't occur in the new training set are used for testing

The bootstrap - example

A dataset of *n* *instances*



Training set: sampling *n* times with replacement

Test set: instances not in the training set

The 0.632 bootstrap

- N instances sampled n times
  - 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:

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

  - 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
  - Training set contains only ~63% of the instances
- Thus it is combined with the resubstitution error:

$$err = 0.632 \cdot e_{\text{test instances}} + 0.368 \cdot e_{\text{training instances}}$$

- The resubstitution error gets less weight than the error on the test data
- Process is repeated several times, with different replacement samples, and the results averaged
- It is probably the best way of estimating performance for very small datasets

Three-way data splits

- Up to now: performance evaluation of a **single model**

- **Now: model selection (and/or model tuning) and performance evaluation of selected/best/tuned model**
  - *Training set*: used for learning
  - *Validation set*: used to select among several trained classifiers
  - *Test set*: used only to assess the performance of a fully-trained classifier
- Why separate test and validation sets?
  - The error rate of the final model on validation data will be biased (smaller than the true error rate) since the validation set is used to select the final model
  - After assessing the final model on the test set, YOU MUST NOT tune the model any further!

Three-way data split - 2

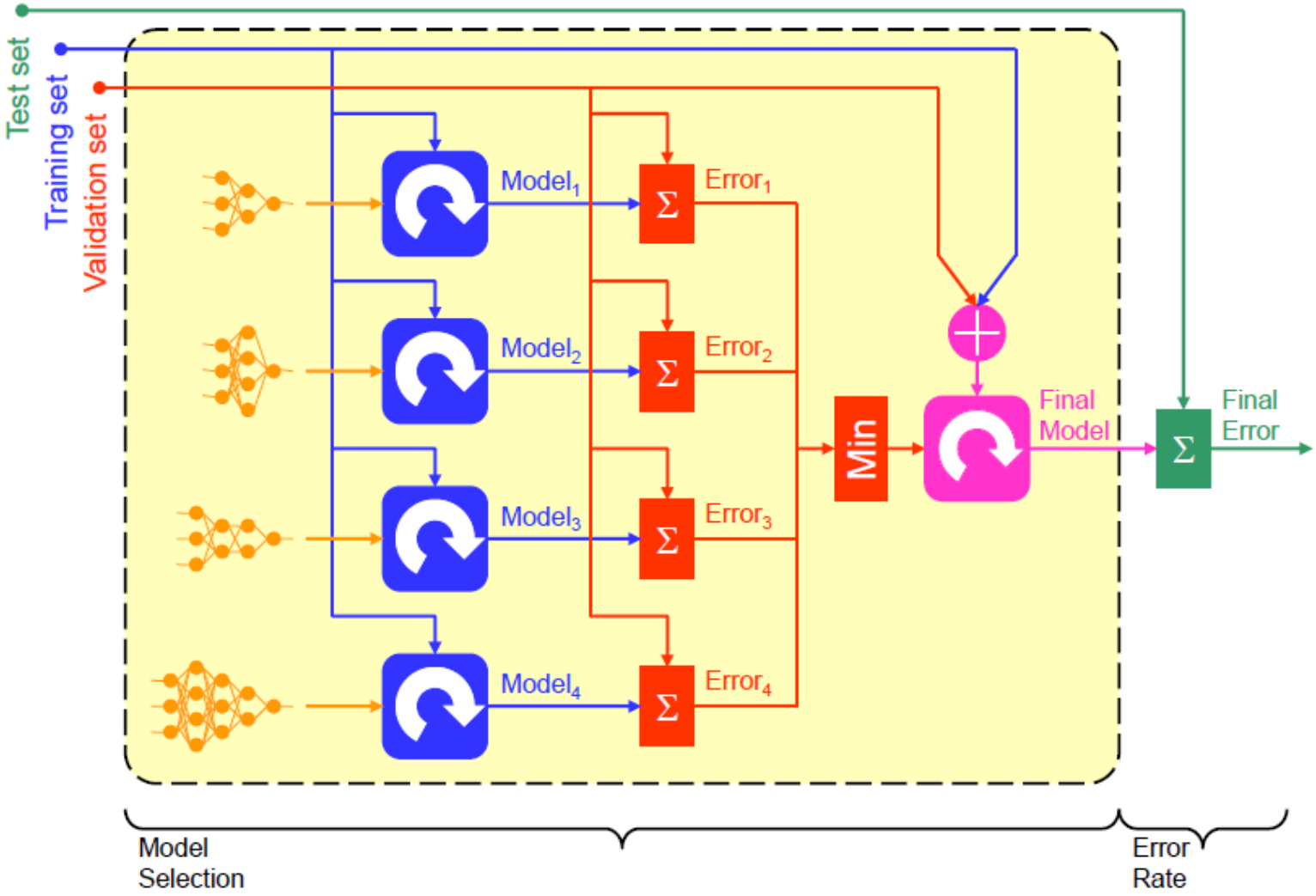
Procedure outline (with holdout method)

1. Divide the available data into training, validation and test set
2. Repeat the following steps with different models(algorithms)/parameters
    1. Select model
    2. Train the model using the training set
    3. Evaluate the model using the validation set

    } Performance evaluation of single model
3. Select the best model and train it using data from the training and validation sets
4. Assess this final model using the test set

If CV or bootstrap are used, steps 2.2 and 2.3 have to be repeated for each of the K folds

Three-way data splits



Competitions: Number of uploads limited, public/private test set

Charts, muptiple values

- More detailed evaluation, diagram
  - Costs are rerely known

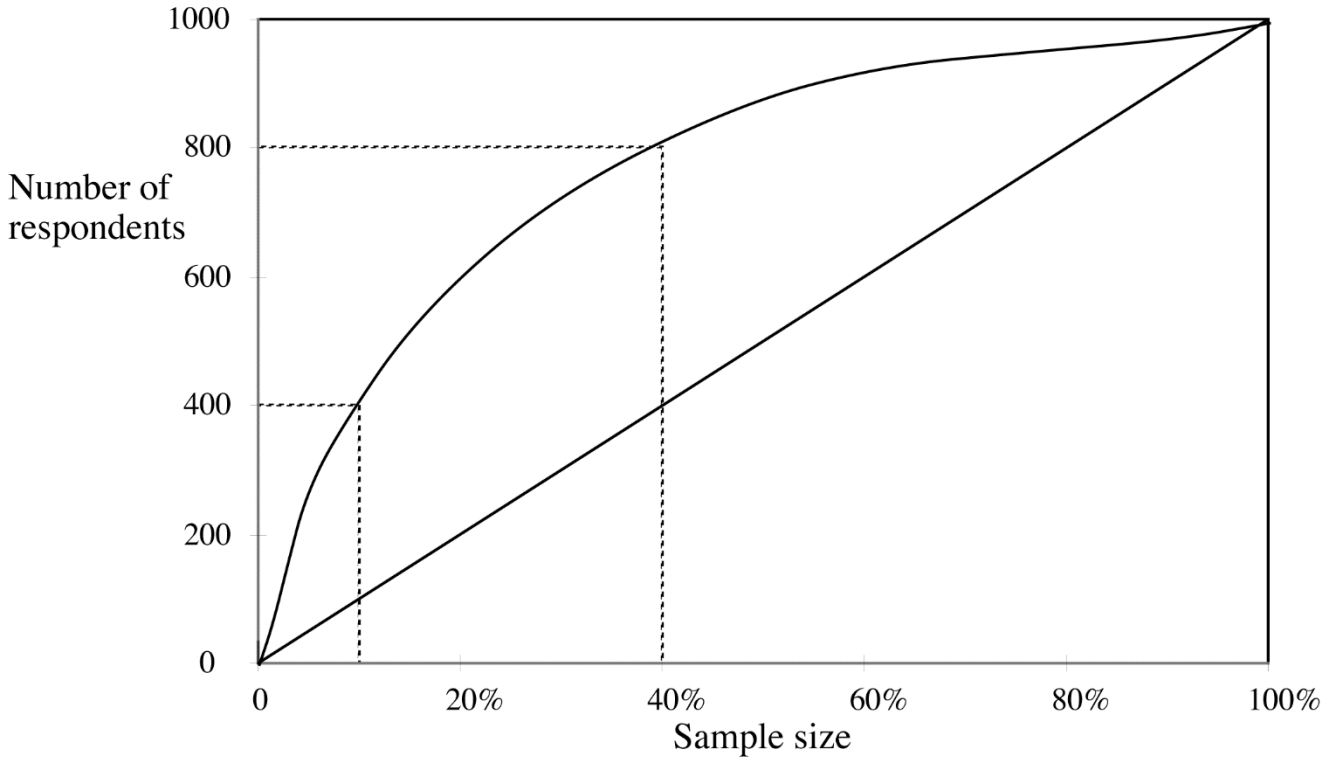
- Case 1: output with some confidence measure
  - Sort instances according to predicted probability of being positive
  - Different result sets by cutting the sorted list at different positions

- Case 2: model with tu

- Measures for result sets in a diagram

Lift chart

number of true positives

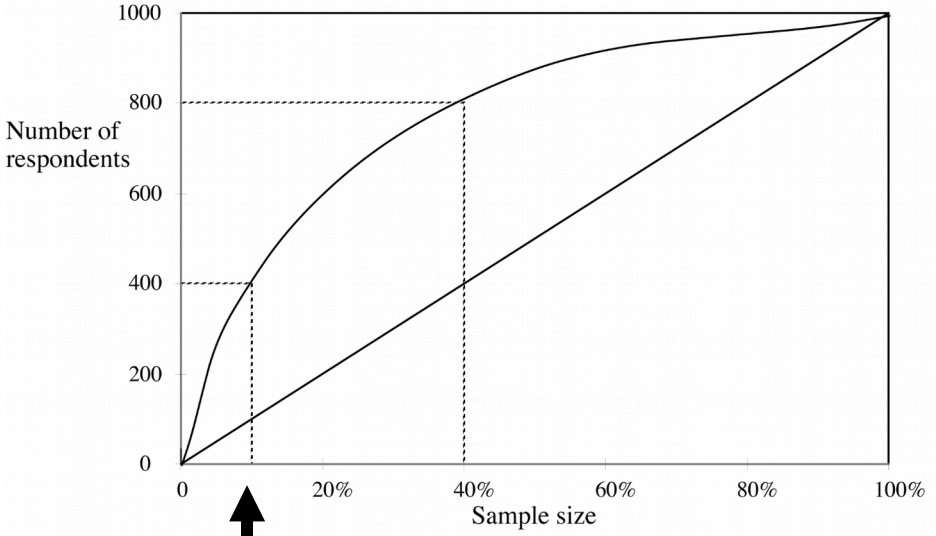


sample size

40% of responses for 10% of cost

80% of responses for 40% of cost

Lift chart: promotional mailing example



400 responses (40%)
for 100 000 mails (10% of costs)

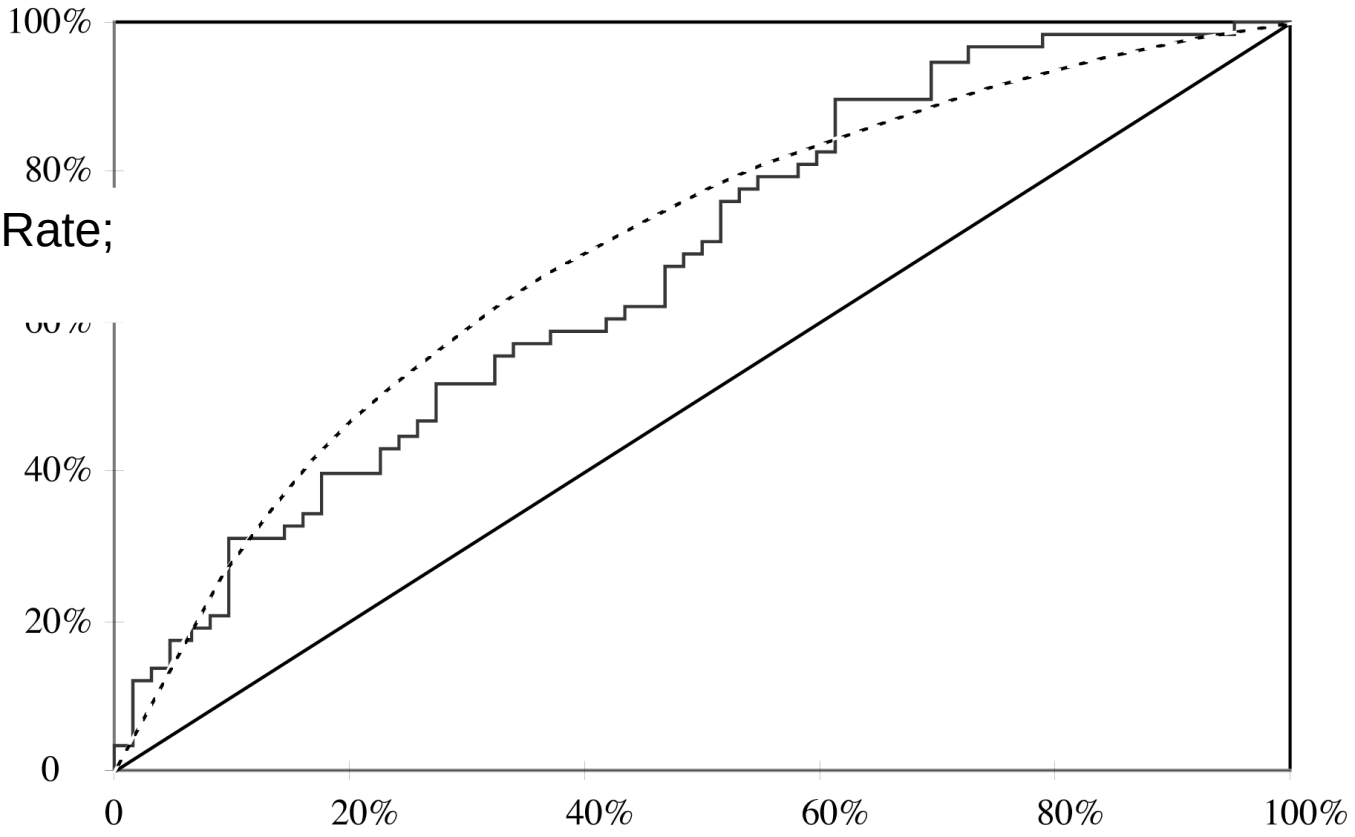
800 responses (80%)
for 400 000 mails (40% of costs)

1000 responses (100%)
for 1 000 000 mails (100% of costs)

ROC („Receive Operation Characteristic”) curves

- *ROC curves* are similar to lift charts
  - Used in signal detection to show tradeoff between hit rate and false alarm rate over noisy channel
- Differences to lift chart:
  - *x* axis
    FPR: False Positive Rate = FP/(FP+TN) =
    =1 –True Negative Rate = 1- Specificity
    (Lift chart: *sample size*)
  - *y* axis
    TPR: True Positive Rate = TP/(TP+FN) = sensitivity
    (Lift chart: absolute number)

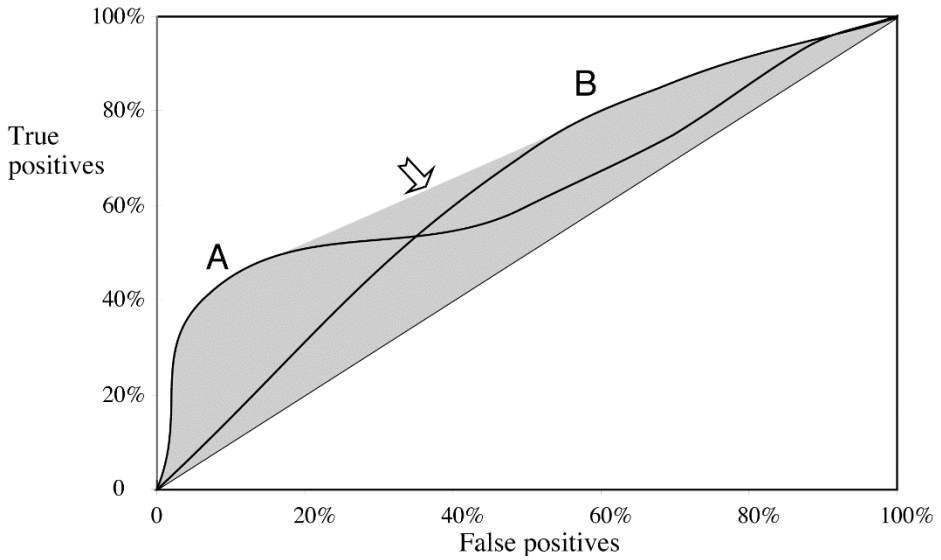
A sample ROC curve



True Positive Rate;
Sensitivity

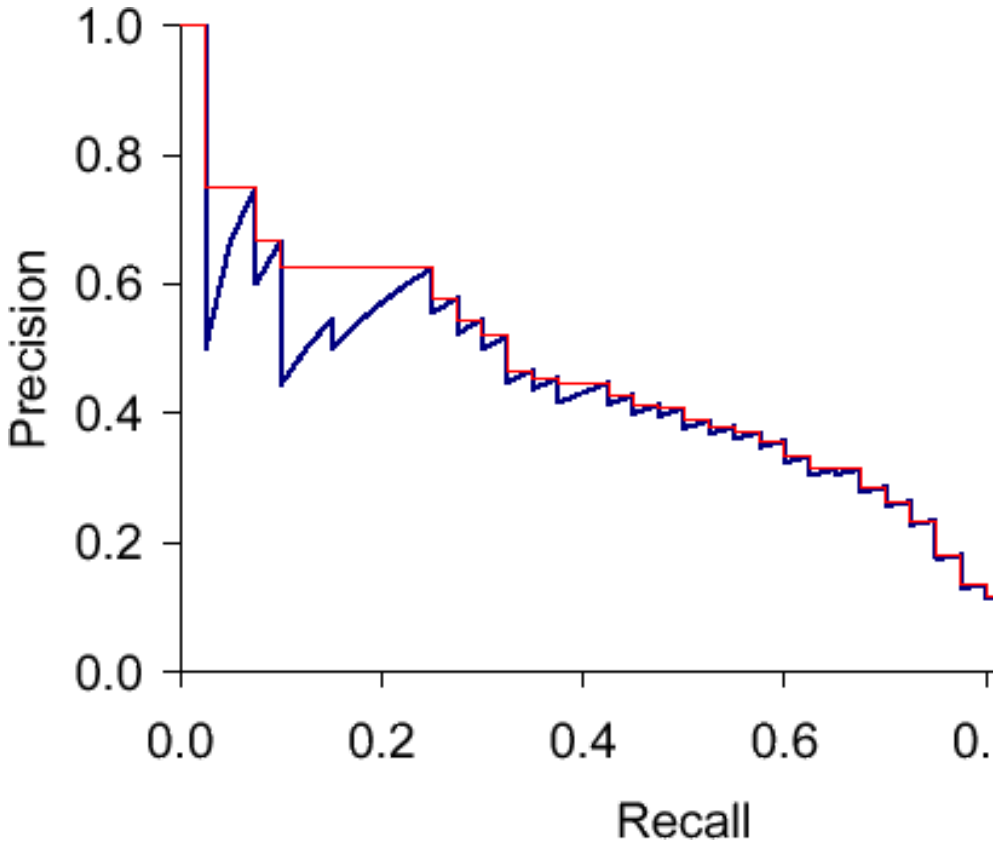
False Positive Rate;
100% - Specificity

Combining ROC curves



- For a small, focused sample, use method A
- For a larger one, use method B
- In between, choose between A and B with appropriate probabilities

Precision-recall curve



Aggregated (summary) measures...

- ***F-measure***=(2 × recall × precision)/(recall+precision) (harmonic mean of precision and recall)
- average precision at 20%, 50% and 80% recall *three-point average recall*
- Specificity * sensitivity
- ***Area under the ROC curve*** (*AUC*): probability that randomly chosen positive instance is ranked above randomly chosen negative one
  - Random: 0.5
  - Ideal: 1
  - Good performance depends on domain

Evaluating numeric prediction

- Same strategies
  - independent test set, cross-validation, significance tests, etc.
- Difference
  - Error measure
- Actual target values: a_1, a_2,…,a_n
- Predicted target values: p_1, p_2,…,p_n
- Most popular measure: *mean-squared error*

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

  - Easy to work with 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* is less sensitive to outliers than the mean-squared error:

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

- Sometimes *relative* error values are more appropriate (e.g. 10% for an error of 50 when predicting 500)

Improvement on the mean

- Often we want to know how much the scheme improves on simply predicting the average (a very simple predictor, $\bar{a}$ is the average).

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

(Relative squared error)

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

(Relative absolute error)

The correlation coefficient

- It measures the *statistical correlation* between the predicted values and the actual values

$$\frac{S_{PA}}{\sqrt{S_P S_A}}$$

$$S_{PA} = \frac{\sum_i (p_i - \bar{p})(a_i - \bar{a})}{n - 1} \qquad S_P = \frac{\sum_i (p_i - \bar{p})^2}{n - 1} \qquad S_A = \frac{\sum_i (a_i - \bar{a})^2}{n - 1}$$

- Scale independent, between –1 and +1

Which measure?

- Best to look at all of them
- Often it doesn't matter
- Example:

	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

Evaluating clustering

- Not trivial! (classification, numeric prediction much more straightforward!)
- Two major types
  - External: additional information on the data
  - Internal: just the data, nothing else

Clustering: external evaluation

- External information source, e.g.
  - classes available
  - human judgement
  - utility of clustering with respect to its application
- Measure (overlapping between clusters and external information)
  - Purity, normalized mutual information, Rand-index, F-measure

Clustering: internal evaluation

- Evaluation based on data that was clustered
- Intra-cluster similarity high
  Inter-cluster similarity low
- When same measure used as for clustering…
  - E.g. K-means with distance-based internal evaluation

Model elegance vs. errors

- Theory 1: very simple, elegant theory that explains the data almost perfectly
- Theory 2: significantly more complex theory that reproduces the data without mistakes

- Which one is better?

Model selection criteria

- Model selection criteria attempt to find a good compromise between:
  - The complexity of a model
  - Its prediction accuracy on the training data
- Reasoning: if no difference in prediction accuracy, simple modell preferred
- Also known as *Occam's Razor* : the best theory is the smallest one that describes all the facts

William of Ockham, born in the village of Ockham in Surrey (England) about 1285, was the most influential philosopher of the 14th century and a controversial theologian.

The MDL principle

- MDL stands for *minimum description length*
- The description length is defined as:

  space required to describe a theory

  +

  space required to describe the theory's mistakes
- In our case the theory is the classifier and the mistakes are the errors on the training data
- Aim: we seek a classifier with minimal DL
- MDL principle is a *model selection criterion*

MDL and compression

- MDL principle relates to data compression:
  - The best theory is the one that compresses the data the most
  - I.e. to compress a dataset we generate a model and then store the model and its mistakes

Summary/concepts/questions

- Overfitting, resubstitution error
- success rate, error rate
- confusion matrix
  - true positive, false positive, true negative, false negative
- Precision, recall, sensitivity, specificity
- training set, test set
- three-way data split: training, validation, test set
- holdout method
- stratification
- cross validation (tenfold, repeated)
  - leave one out cross validation
- bootstrap

Summary/concepts/questions - 2

- Predicting probabilities
  - quadratic loss function, informational loss function
- Considering costs
  - cost matrix
  - cost-based prediction
- lift chart, ROC, Precision-recall curve
- F-measure, Area under ROC

Summary/concepts/questions - 3

- Root mean-squared error
- Mean absolute error
- Correlation coefficient
- Evaluating clustering
  - External
  - internal
- MDL principle