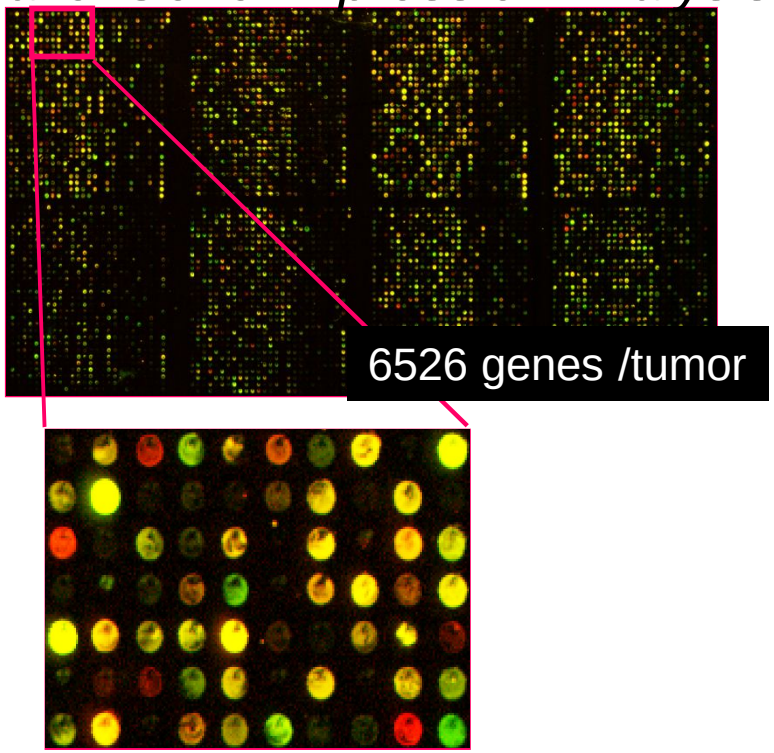
Classification in Microarray Data

Motivation: A study of gene expression on breast tumours (NHGRI, J. Trent)

cDNA Microarrays

Parallel Gene Expression Analysis



- How similar are the gene expression profiles of BRCA1 and BRCA2 (+) and sporadic breast cancer patient biopsies?
- ***Can we identify a set of genes that distinguish the different tumor types?***
- Tumors studied:
  - 7 BRCA1 +
  - 8 BRCA2 +
  - 7 Sporadic

Classification

- A predictor or classifier for K [tumor] classes partitions the space X of gene expression profiles into K disjoint subsets, $A_1, ..., A_K$, such that for a sample with expression profile $x=(x_1, ...,x_p) \in A_k$ the predicted class is k.

- Predictors are built from past experience, i.e., from observations which are known to belong to certain classes. Such observations comprise the Training set

$$L = (x_1, y_1), ..., (x_n,y_n).$$

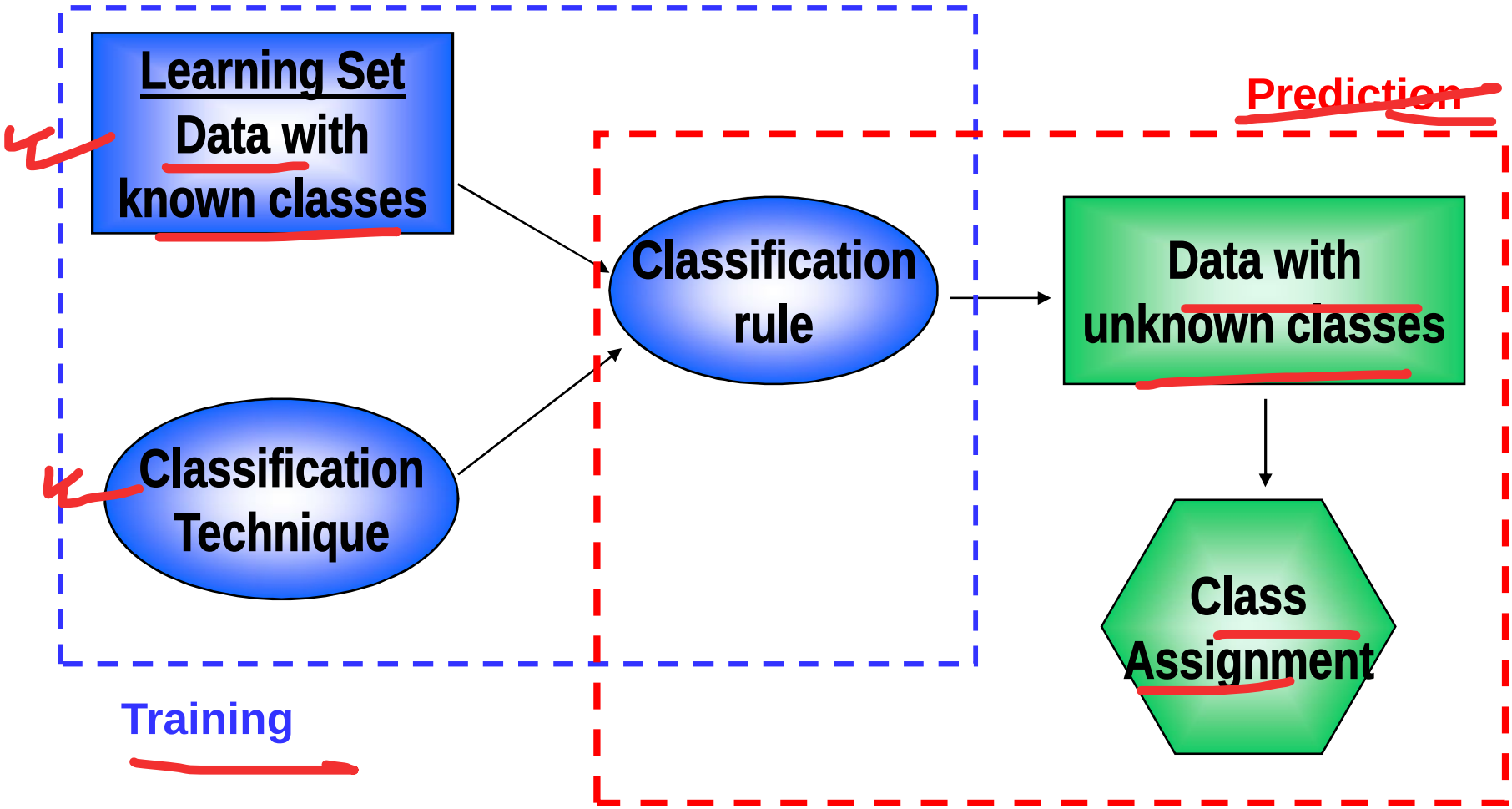
- A classifier built from a Training set L is denoted by

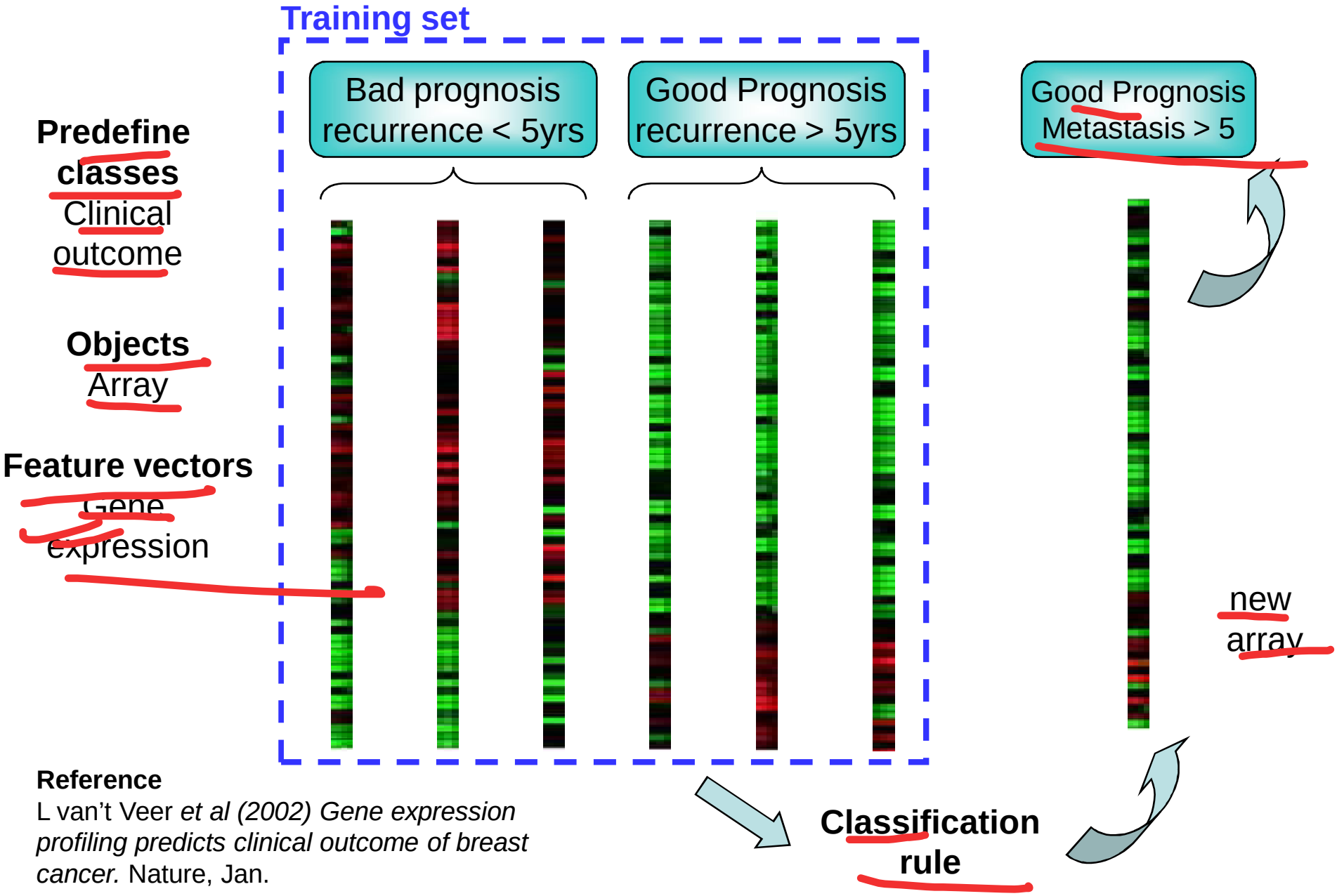
$$C(. ,L): X \rightarrow \{1,2, ... ,K\},$$

with the predicted class for observation x being $C(x,L)$.

Supervised learning

Classification and Allocation

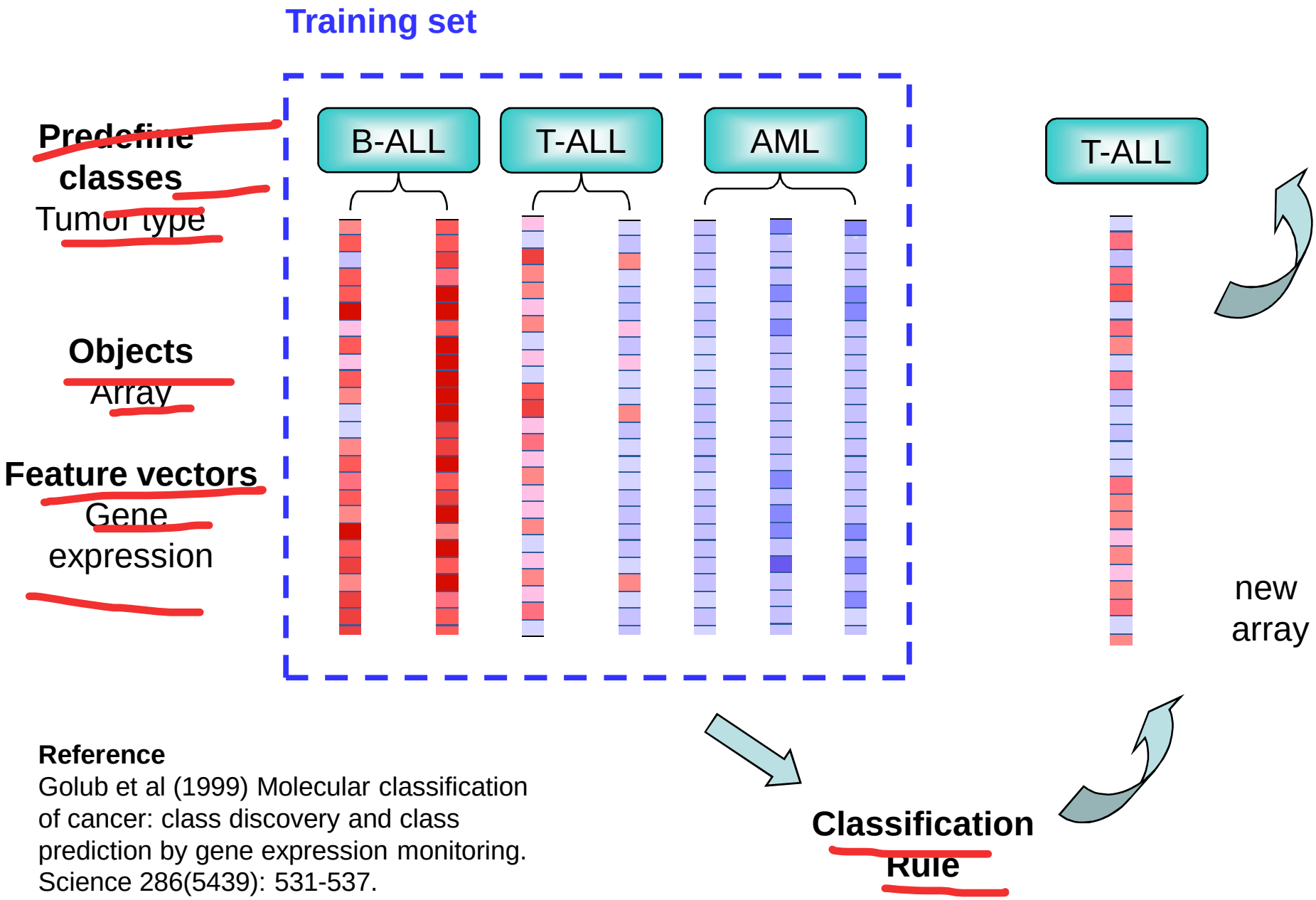




Reference

L van't Veer *et al (2002) Gene expression profiling predicts clinical outcome of breast cancer.* Nature, Jan.

.



Reference
Golub et al (1999) Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. Science 286(5439): 531-537.

Components of class prediction

- Choose a method of class prediction
  - LDA, KNN, CART, ...: Prediction model
- Select genes on which the prediction will be base: Feature selection
  - Which genes will be included in the model?
- Validate the model
  - Use data that have not been used to fit the predictor

Prediction methods

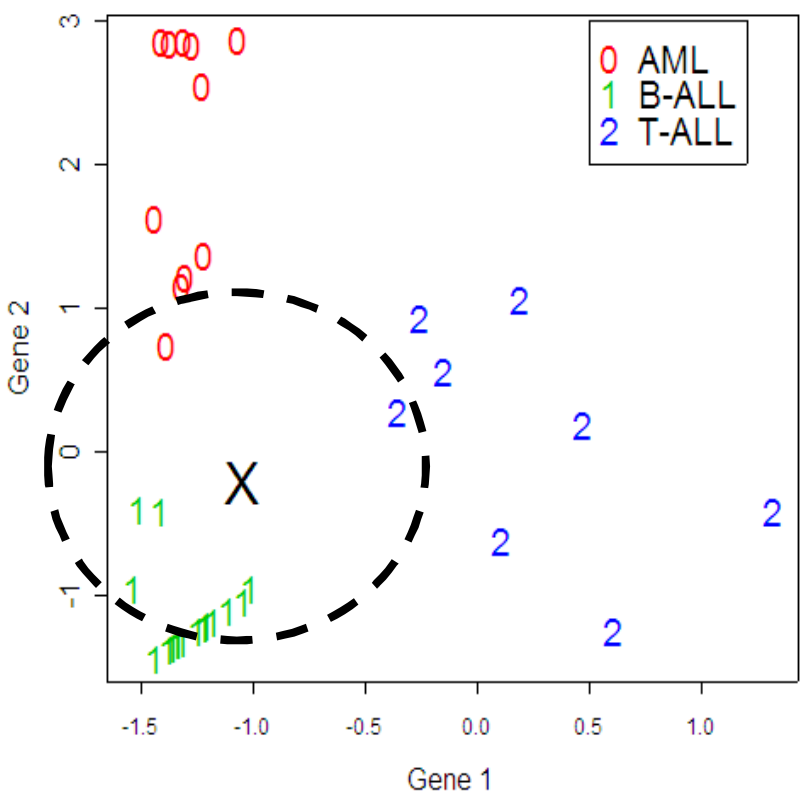
Choose prediction model

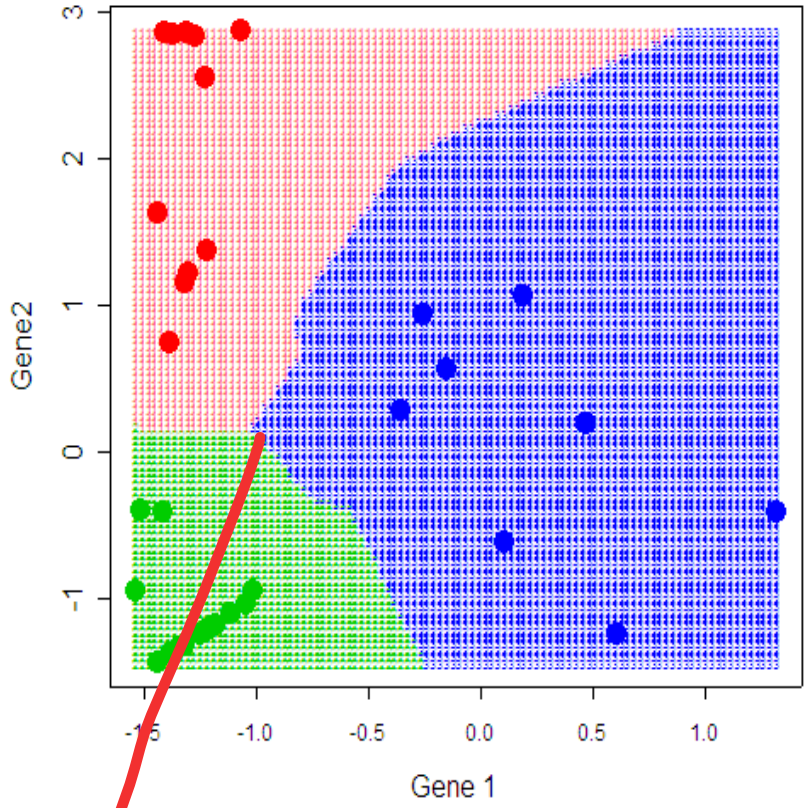
- Prediction methods
  - Fisher linear discriminant analysis (FLDA) and its variants (DLDA, Gene voting, CCP, ...)
  - Logistic classification
  - K-Nearest Neighbor
  - Classification Trees
  - Support vector machines (SVMs)
  - Neural networks
  - And many more …

Nearest neighbor classification

- Based on a measure of distance between observations (e.g. Euclidean distance or one minus correlation).
- **k-nearest neighbor rule** (Fix and Hodges (1951)) classifies an observation ***x*** as follows:
  - find the ***k*** observations in the learning set **closest** to **x**
  - predict the class of **x** by **majority vote**, i.e., choose the class that is most common among those k observations.
- The number of neighbors k can be chosen by **cross-validation** (more on this later).

Nearest neighbor rule





decision boundary

Other classifiers include…

- Support vector machines
- Neural networks
- Bayesian regression methods
- Projection pursuit
- ….

Feature selection

Feature selection

- A classification rule must be based on a set of variables which contribute useful information for distinguishing the classes.
- This set will usually be small because most variables are likely to be uninformative.
- Some classifiers (like CART) perform automatic feature selection whereas others, like LDA or KNN, do not.

Approaches to feature selection

- **Filter methods** perform *explicit* feature selection prior to building the classifier.
  - One gene at a time: select features based on the value of an univariate test.
  - The number of genes or the test p-value are the parameters of the FS method.
- **Wrapper methods** perform FS *implicitly*, as a part of the classifier building.
  - In classification trees features are selected at each step based on reduction in impurity.
  - The number of features is determined by pruning the tree using cross-validation.

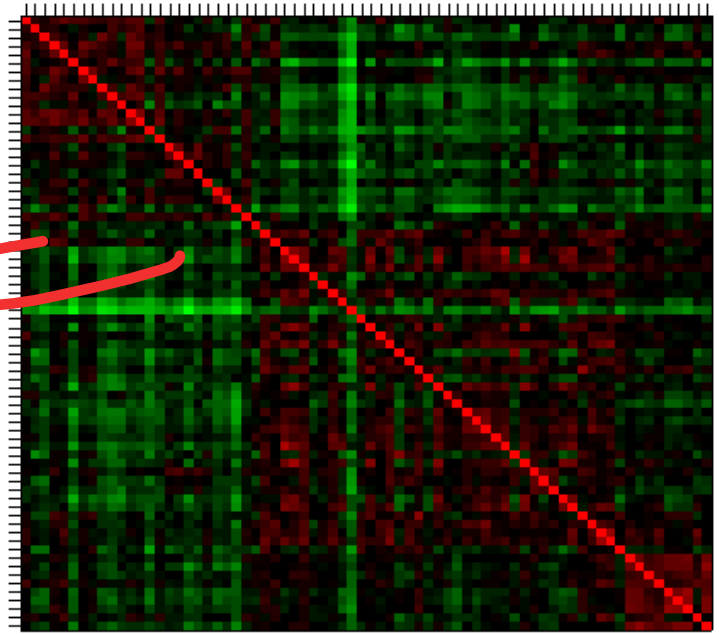
we select one feature and try to predict, is it helpful or not # checking we don't want multiple feature/column

like in Decision Tree

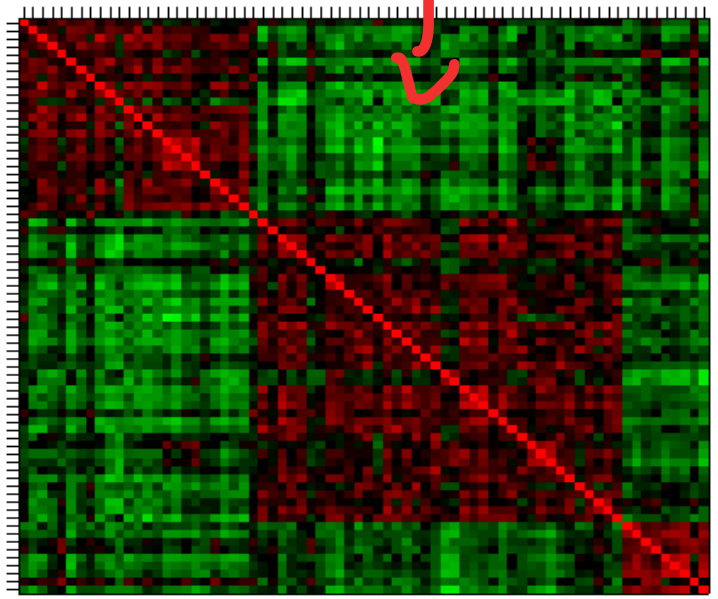
Why select features

- Lead to better classification performance by removing variables that are noise with respect to the outcome
- May provide useful insights into etiology of a disease.
- Can eventually lead to the diagnostic tests (e.g., “breast cancer chip”).

Why select features?



No feature selection



Top 100
feature selection
Selection based on variance

Correlation plot

Data: Leukemia, 3 class

Performance assessment

Performance assessment

- Before using a classifier for prediction or prognostic one needs a measure of its accuracy.
- The accuracy of a predictor is usually measured by the Missclassification rate: *The % of individuals belonging to a class which are erroneously assigned to another class by the predictor.*
- An important problem arises here
  - We are not interested in the ability of the predictor for classifying current samples
  - One needs to estimate *future performance* based on *what is available.*

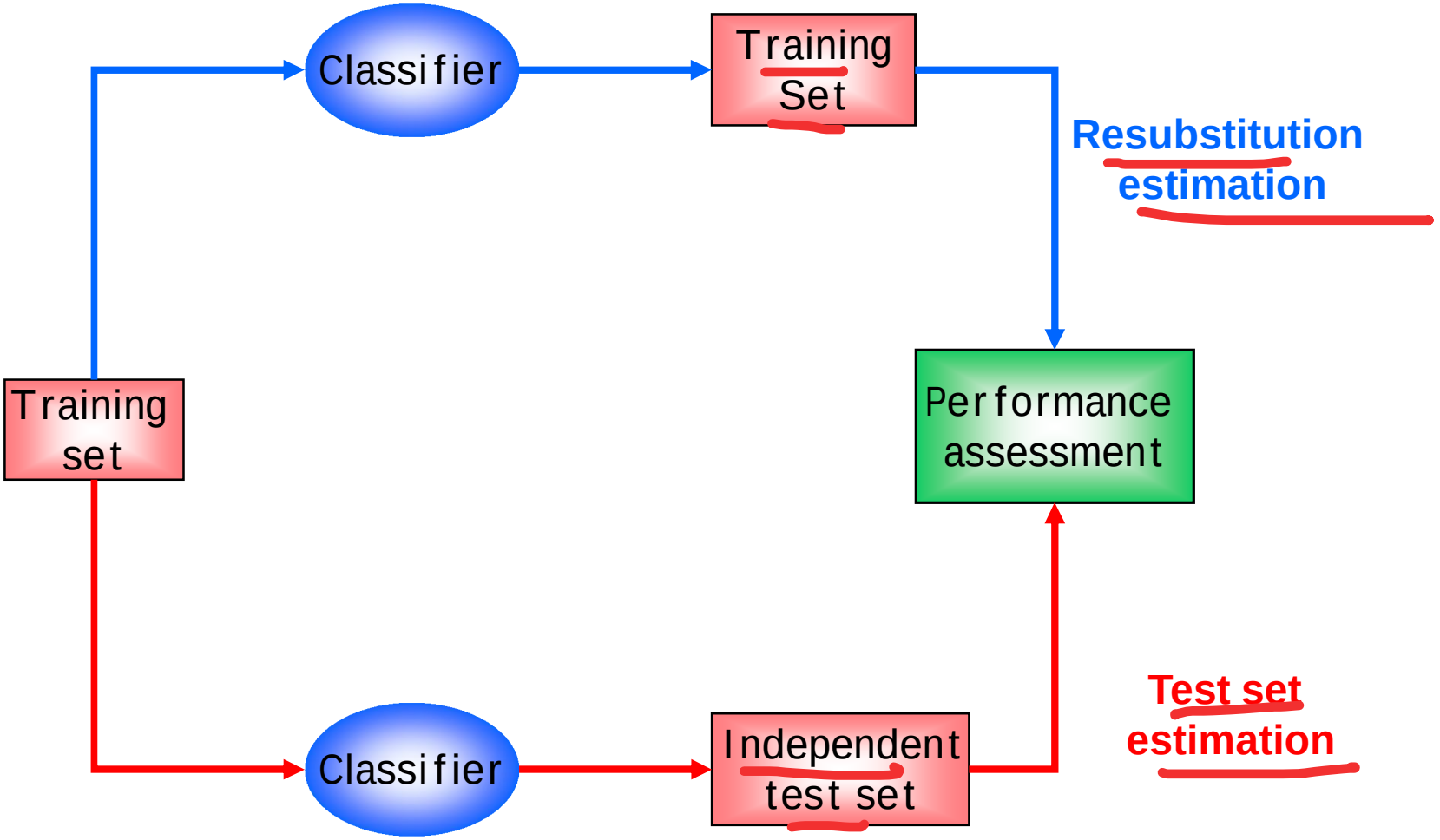
Estimating the error rate

- Using the same dataset on which we have built the predictor to estimate the missclassification rate may lead to erroneously low values due to overfitting.
  - This is known as the ***resubstitution estimator***
- We should use a completely independent dataset to evaluate the classifier, but it is rarely available.
- We use alternatives approaches such as
  - ***Test set estimator***
  - ***Cross validation***

Performance assessment (I)

- Resubstitution estimation: Compute the error rate on the learning set.
  - Problem: downward bias
- Test set estimation: Proceeds in two steps
  1. Divide learning set into two sub-sets, L and T;
  2. Build the classifier on L and compute error rate on T.
  - This approach is not free from problems
    - L and T must be independent and identically distributed.
    - Problem: reduced effective sample size

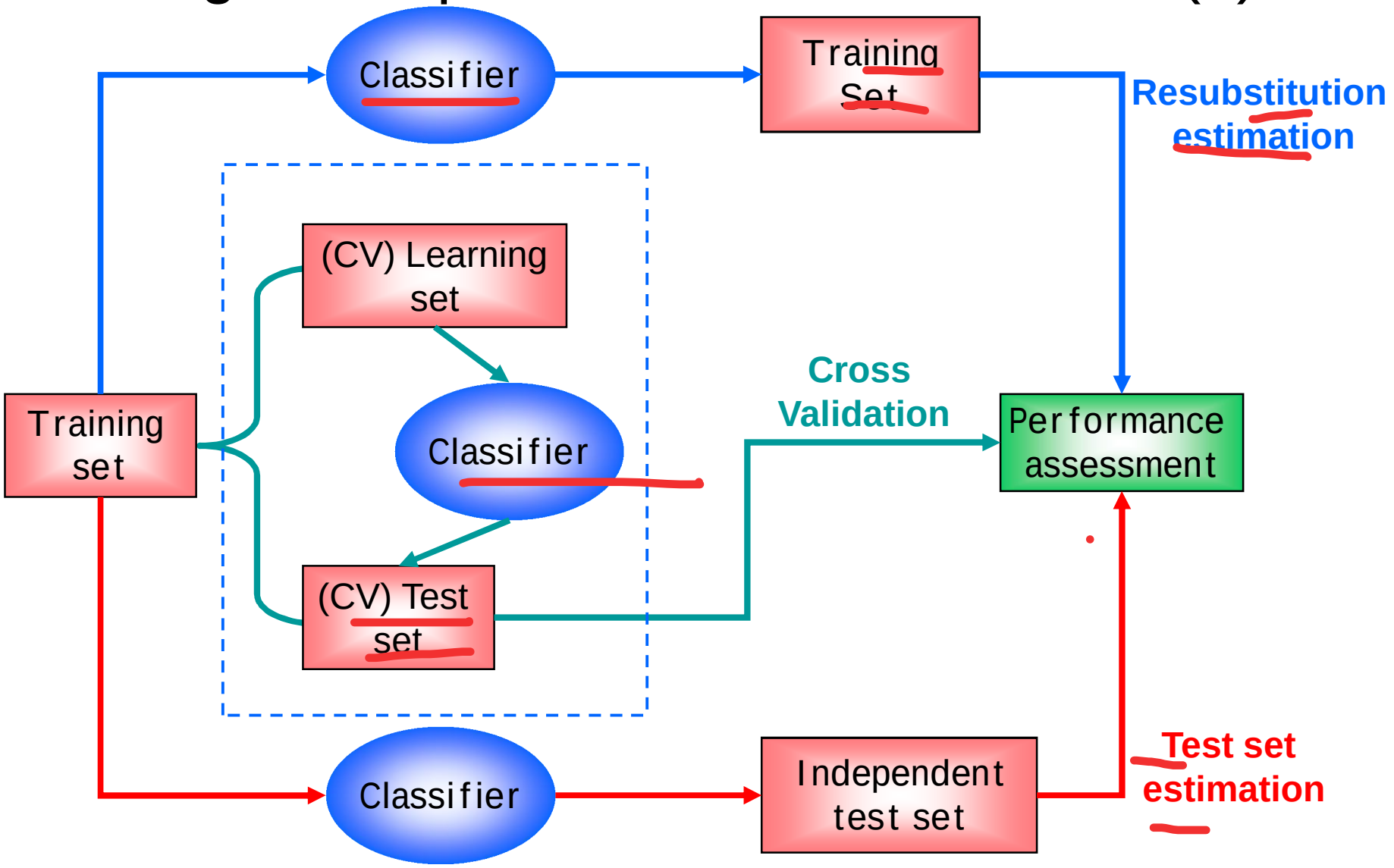
Diagram of performance assessment (I)



Performance assessment (II)

- V-fold cross-validation (CV) estimation: Cases in learning set randomly divided into V subsets of (nearly) equal size. Build classifiers by leaving one set out; compute test set error rates on the left out set and averaged.
  - Bias-variance tradeoff: smaller V can give larger bias but smaller variance
  - Computationally intensive.
- Leave-one-out cross validation (LOOCV).
  - Special case for V=n.
  - Works well for stable classifiers (k-NN, LDA, SVM)

Diagram of performance assessment (II)



Performance assessment (III)

- Common practice
  - To do feature selection using the learning,
  - To do CV only for model building and classification.
- However, usually features are unknown and the intended inference includes feature selection →

  CV estimates as above tend to be downward biased.
- Features (variables) should be selected only from the learning set used to build the model (and not the entire set).
- Classification accuracy may be misleading for imbalanced classes.

Classifier Evaluation Metrics: Confusion Matrix

Confusion Matrix:

Actual class\Predicted class →	C_1	$\neg C_1$
C_1 ↓	**True Positives (TP)**	**False Negatives (FN)**
$\neg C_1$	**False Positives (FP)**	**True Negatives (TN)**

Example of Confusion Matrix:

Actual class\Predicted class	buy_computer = yes	buy_computer = no	Total
buy_computer = yes	**6954**	**46**	7000
buy_computer = no	**412**	**2588**	3000
Total	7366	2634	10000

- Given *m* classes, an entry, $CM_{i,j}$ in a **confusion matrix** indicates # of tuples in class *i* that were labeled by the classifier as class *j*
- May have extra rows/columns to provide totals

Classifier Evaluation Metrics: Accuracy, Error Rate, Sensitivity and Specificity

A\P	C	¬C	
C	**TP**	**FN**	**P**
¬C	**FP**	**TN**	**N**
	P'	**N'**	**All**

Classifier Accuracy, or recognition rate: percentage of test set tuples that are correctly classified

Accuracy = (TP+TN)/(TP+TN+FP+FN)

Error rate: *1 – accuracy*, or

Error rate = (FP+FN)/(TP+TN+FP+FN)

- **Class Imbalance Problem**:
  - One class may be *rare*, e.g. fraud, or HIV-positive
  - Significant *majority of the negative class* and minority of the positive class
  - **Sensitivity**: True Positive recognition rate
    - **Sensitivity = TP/(TP+FN)**
  - **Specificity**: True Negative recognition rate
    - **Specificity = TN/(TN+FP)**

Classifier Evaluation Metrics: Precision and Recall, and F-measures

- **Precision**: exactness – what % of tuples that the classifier labeled as positive are actually positive

$$precision = \frac{TP}{TP + FP}$$

- **Recall:** completeness – what % of positive tuples did the classifier label as positive?

$$recall = \frac{TP}{TP + FN}$$

- Perfect score is 1.0

- Inverse relationship between precision & recall

- ***F* measure (*F_1* or *F*-score)**: harmonic mean of precision and recall,

$$F = \frac{2 \times precision \times recall}{precision + recall}$$

Classifier Evaluation Metrics: Example

Actual Class\Predicted class	cancer = yes	cancer = no	Total	Recognition(%)
cancer = yes	**90**	**210**	300	30.00 (*sensitivity*
cancer = no	**140**	**9560**	9700	98.56 (*specificity)*
Total	230	9770	10000	96.40 (*accuracy*)

- *Precision* = 90/230 = 39.13%
- *Recall* = 90/300 = 30.00%

Evaluating Classifier Accuracy: Holdout & Cross-Validation Methods

- **Holdout method**
  - Given data is randomly partitioned into two independent sets
    - Training set (e.g., 2/3) for model construction
    - Test set (e.g., 1/3) for accuracy estimation
  - Random sampling: a variation of holdout
    - Repeat holdout k times, accuracy = avg. of the accuracies obtained
- **Cross-validation** (*v*-fold, where v = 10 is most popular)
  - Randomly partition the data into *v mutually exclusive* subsets, {D1,D2,…Dv} each approximately of equal size
  - At *i*-th iteration, use D_i as test set and others as training set
  - Leave-one-out: *v* folds where *v* = no. of tuples, for small sized data
  - ***Stratified cross-validation***: folds are stratified so that class distrtibution in each fold is approx. the same as that in the initial data

Evaluating Classifier Accuracy: Bootstrap

- **Bootstrap**
  - Works well with small data sets
  - Samples the given training tuples uniformly *with replacement*
    - i.e., each time a tuple is selected, it is equally likely to be selected again and re-added to the training set
- Several bootstrap methods, and a common one is **.632 boostrap**
  - A data set with *d* tuples is sampled *d* times, with replacement, resulting in a training set of *d* samples. The data tuples that did not make it into the training set end up forming the test set. About 63.2% of the original data end up in the bootstrap, and the remaining 36.8% form the test set (since $(1 – 1/d)^d \approx e^{-1} = 0.368$)
  - Repeat the sampling procedure *k* times to find overall accuracy of the model: