COMPUTER SCIENCE
INDIANA UNIVERSITY
School of Informatics and Computing
Bloomington

Performance measures

Crash course in evaluation

- Running a scientific experiment to compare machine learning algorithms necessary to draw conclusions
- Needs to be meticulous, even if sometimes tedious or need to re-run experiments
- Requires an experiment design and statistical significance tests

Experimental set-up

- Performance measures
- Sampling: How to obtain multiple samples of performance?
- Making conclusions: Statistical significance tests
- Careful statistical work done on executing empirical studies; pros and cons to each
  - for a nice reference, see Evaluating Learning Algorithms: A Classification Perspective (http://www.mohakshah.com/tutorials/icml2012/Tutorial-ICML2012/Tutorial_at_ICML_2012.html); slides in this lecture use some of the material there
  - "Prediction error estimation: a comparison of resampling methods"

Statistical significant test

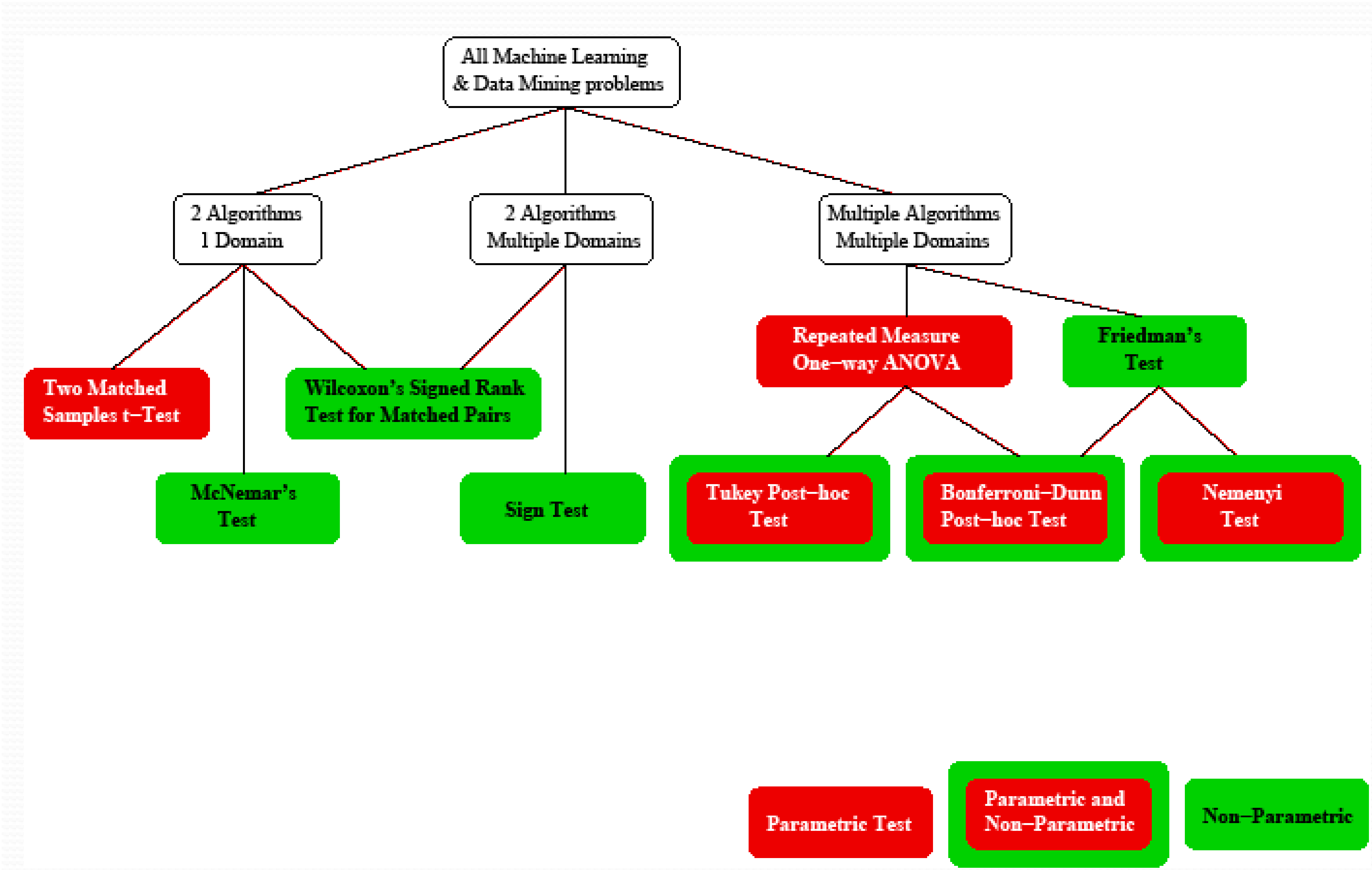
- Can the observed results be attributed to real characteristics of the learner under scrutiny or are they observed by chance?

- Hypothesis testing:
  - State a null hypothesis, e.g., the expected errors of two classifiers is equivalent
  - Choose a statistical significance test to reject the null hypothesis; failing to reject the null hypothesis does not mean we accept it
  - Rejecting the null hypothesis gives us some confidence in the belief that our observations did not occur merely by chance.

How to choose tests?

- Try to satisfy assumptions and use some rules of thumb
- Parametric statistical tests make stronger assumptions about the distribution of the data
- Non-parametric tests make weaker assumptions, but are less powerful (less able to reject the null hypothesis when it is false)
- Selection based on type of problem
  - comparing 2 algorithms on a single domain
  - comparing 2 algorithms across domains
  - comparing multiple algorithms across domains

Statistical test summary

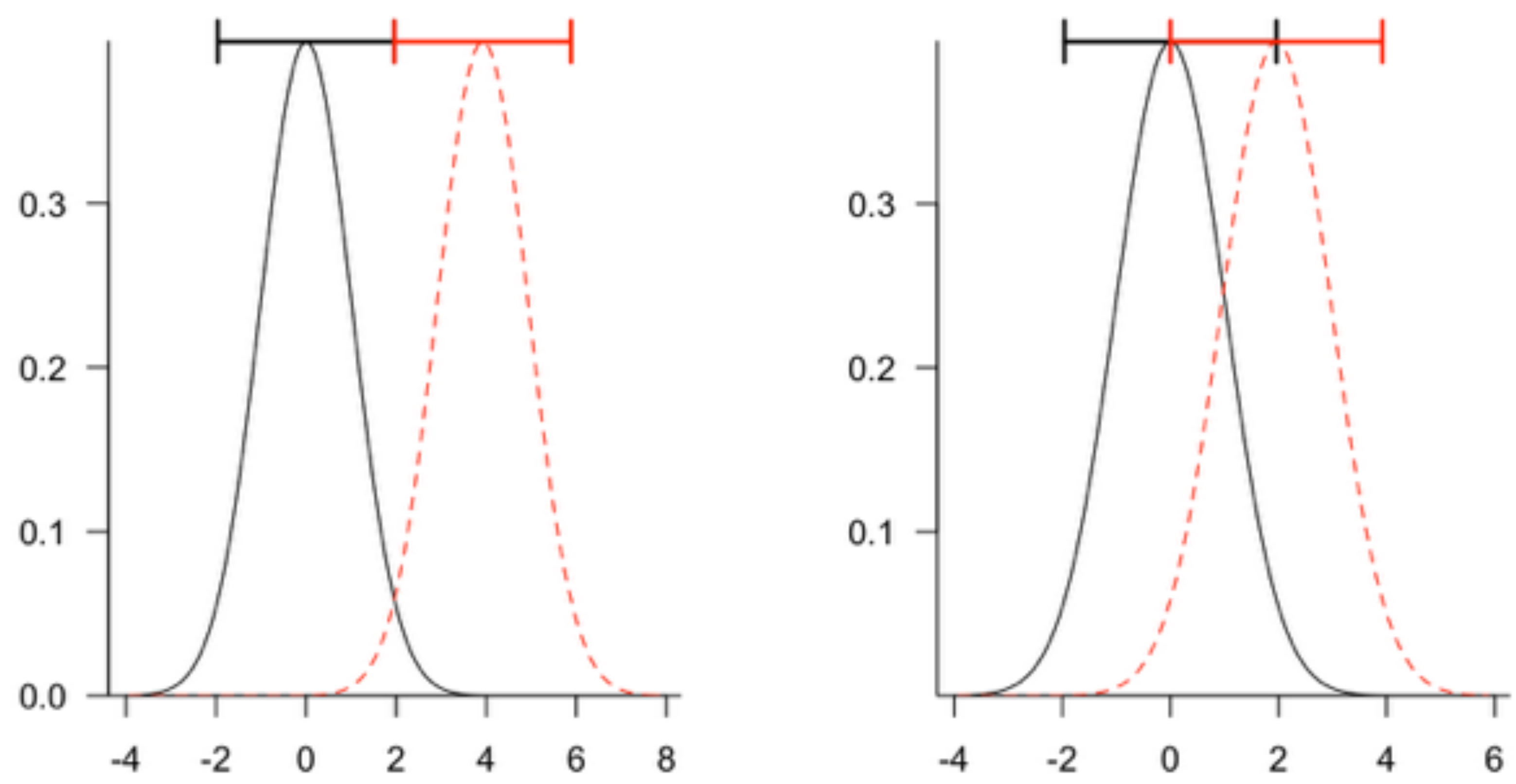


Comparing two algorithms on a single domain

- Imagine you have N independent test-samples, giving N paired measures of error for the two algorithms

- Simplest (not very powerful) strategy:
  - compute two (95%) confidence intervals for the means
  - if the two intervals do not overlap, means are significantly different

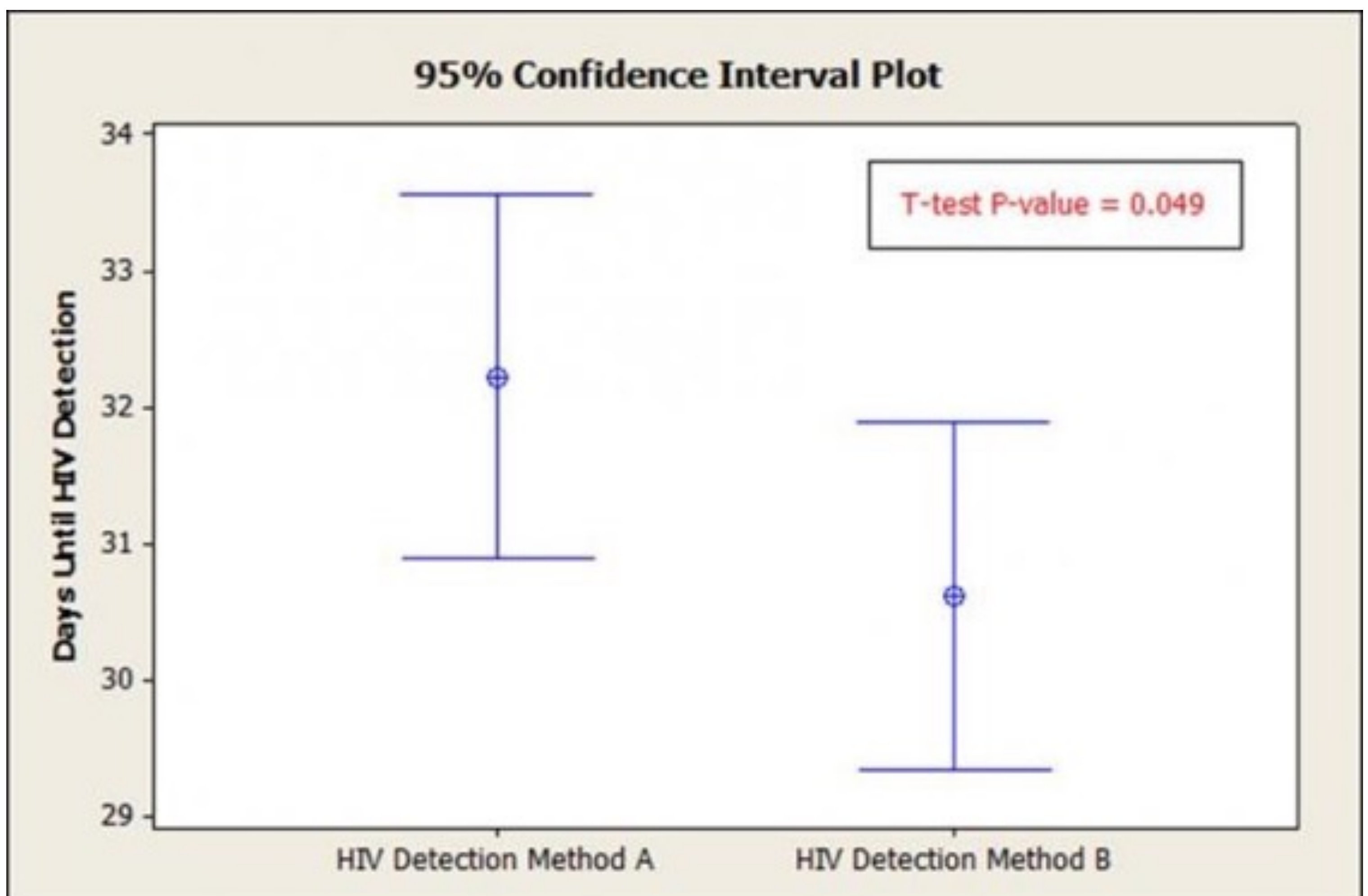
different



unknown

More powerful strategy: t-test

- Confidence intervals may overlap, but the means may still be statistically different
- Paired t-test enables a more powerful comparison
  - more ability to reject the null hypothesis



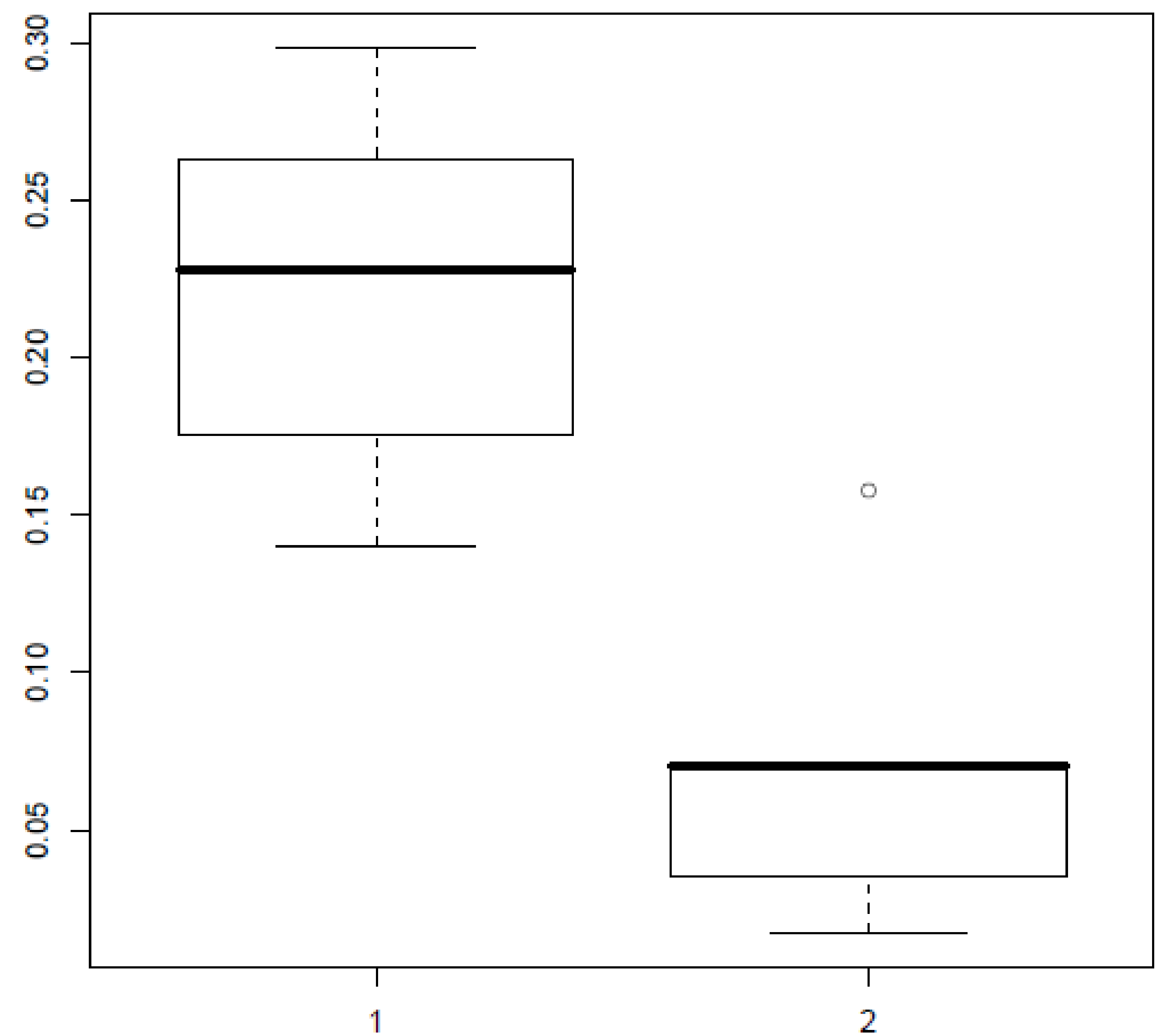
* image from http://blog.minitab.com/blog/real-world-quality-improvement/common-statistical-mistakes-you-should-avoid

Assumptions of the t-test

- The Normality or Pseudo-Normality Assumption: samples come from normally distributed population. Alternatively, the sample size of the testing set should be greater than 30.
- The Randomness of the Samples: The sample should be representative of the underlying population. Therefore, the instances of the testing set should be randomly chosen from their underlying distribution.
- Equal Variance of the populations: The two samples come from populations with equal variance.

Example where assumptions of t-test violated

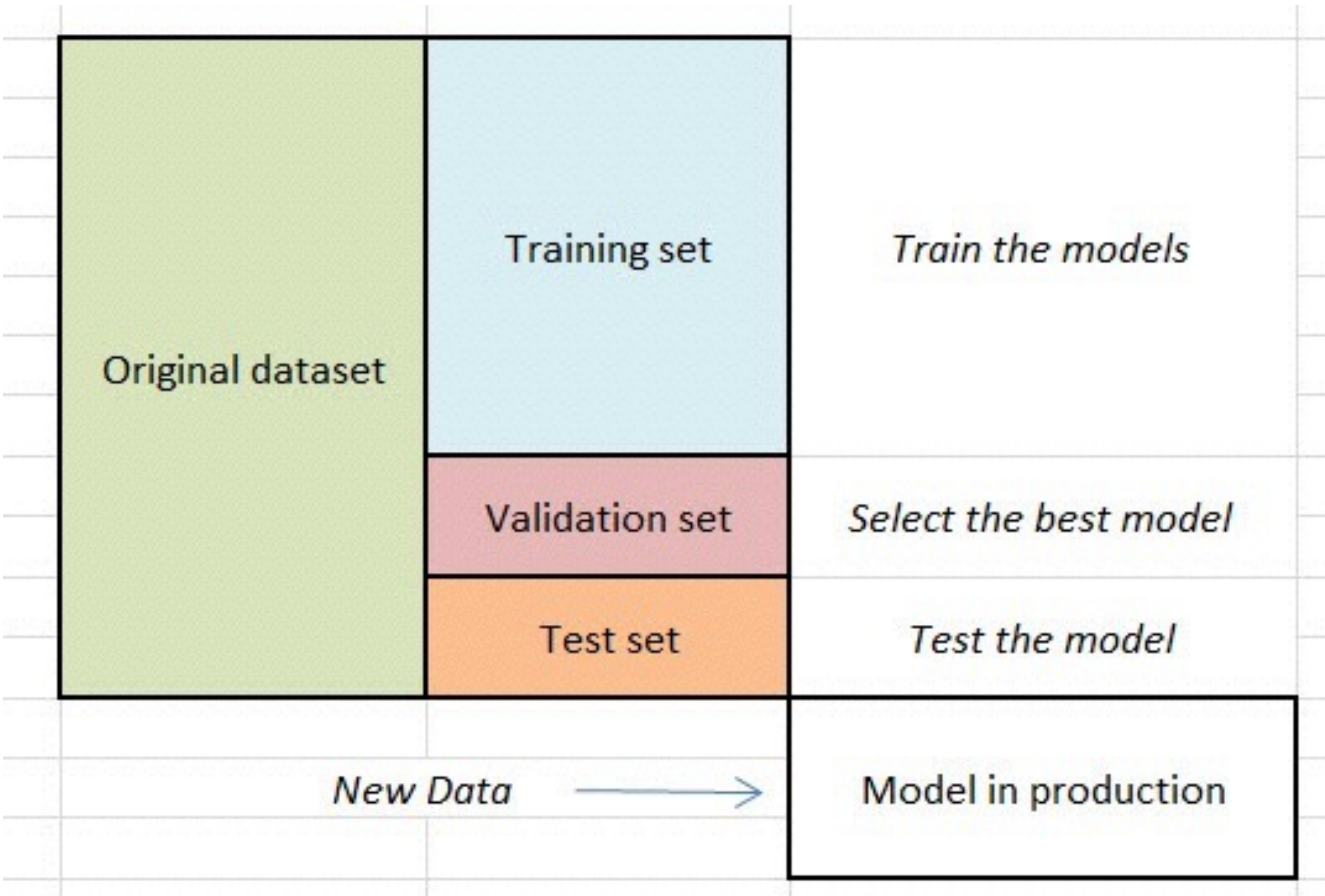
- Equal Variance: variance of C4.5 and NB cannot be considered equal.
- Not warranted to use the t-test to compare C4.5 to NB on the Labour data.
- A better test to use is the non-parametric alternative to the t-test: McNemar's Test



Sampling

- How do we get independent samples of test error? And why?
  - want to get a measure of generalization performance for an algorithm
  - want to compare algorithms
  - want to do hyperparameter selection (e.g., regularization parameters)
- Hold-out test set
- Subsampling approaches
  - Cross-validation (CV)
  - Repeated subsampling: Monte carlo CV
  - Bootstrap resampling

Train-Validation-Test



With lots of data…

- Can afford to have a large hold-out test set that is never used for training
- Data split into train and split
  - Training set may be further split into train and validation, to provide a means to select hyperparameters
- Even with only one split, with a large amount of data likely to get a reasonable estimate of generalization error
- But this approach is error prone; can be difficult to ensure that did not accidentally peak at the data

Thought exercise

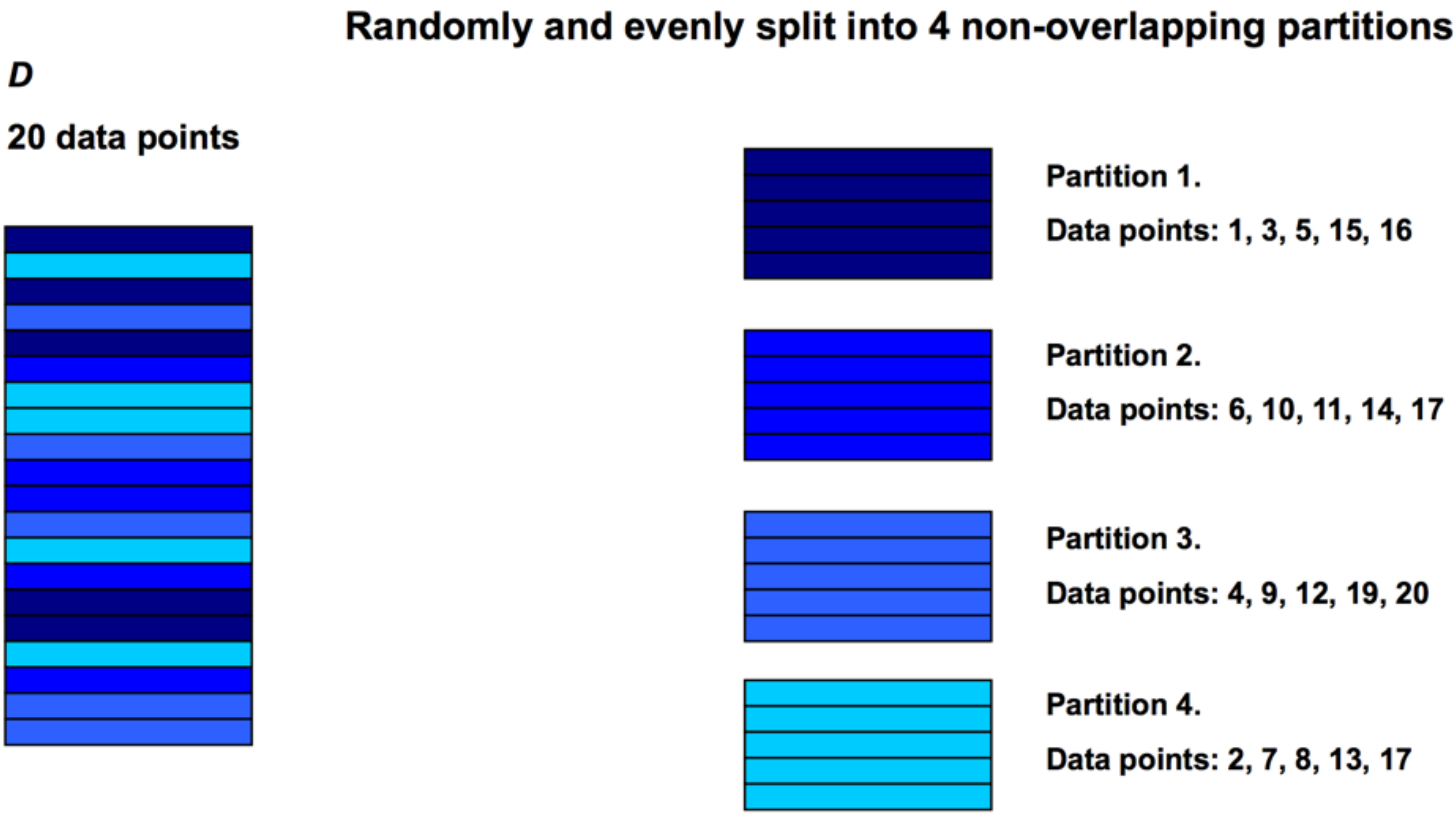
- Imagine you learn on the training set for a fixed hyper parameter setting, and get an error measure on the validation set. Do you expect this to overestimate or underestimate your generalization error?

- Imagine you
  - pick a regularization weight using the validation set
  - then you check your test error and notice it is low
  - then you expand your range of regularization weights to improve performance
  - Do you expect the error on the validation set to be an overestimate or underestimate of the generalization error?
  - What about the test error?

Monte carlo CV

- Also called “repeated learning testing-model”
- Randomly sample without replacement the training set and the test set
  - or for smaller datasets, first sample the training set and use the rest for test
- Repeat this random subsample m times to obtain m training/ test splits

k-fold CV



- Learn model on k-1 folds and test on the hold-out fold (done k times); average k error estimates

Internal and External CV

- Internal CV used to select hyperparameters
  - this can be thought of as part of the algorithm, not actually used to evaluate the differences
  - let data decide what hyperparameters to use, so need reasonable subsampling approach to evaluate selection
- External CV used to obtain performance measures
  - e.g., properly compare two algorithms

Example

- For training set i, test hyperparameters using (internal) k-fold CV
- Select the "best" hyperparameters according to average error
- Given these "best" parameters, learn on entire training set i
- Obtain test error on test set i
- This gives you an estimate of error for training/test split i
- Over all m splits, get m estimates of error (different from k)

Which sampling approach should I use?

- No definitive answer, mostly empirical support
- For how to select k, bias-variance trade-off
  - for small k, high-bias and low-variance
  - for large k, low bias but high variance (e.g., leave-one-out)
  - Some experiments showing that a reasonable balance is k = 10
  - Also determined by computational resources; large k expensive
- For how to select between sampling methods,
  - repeated CV and Monte Carlo CV shown to have fewer Type 1 errors
- Criteria for internal and external CV may be quite different

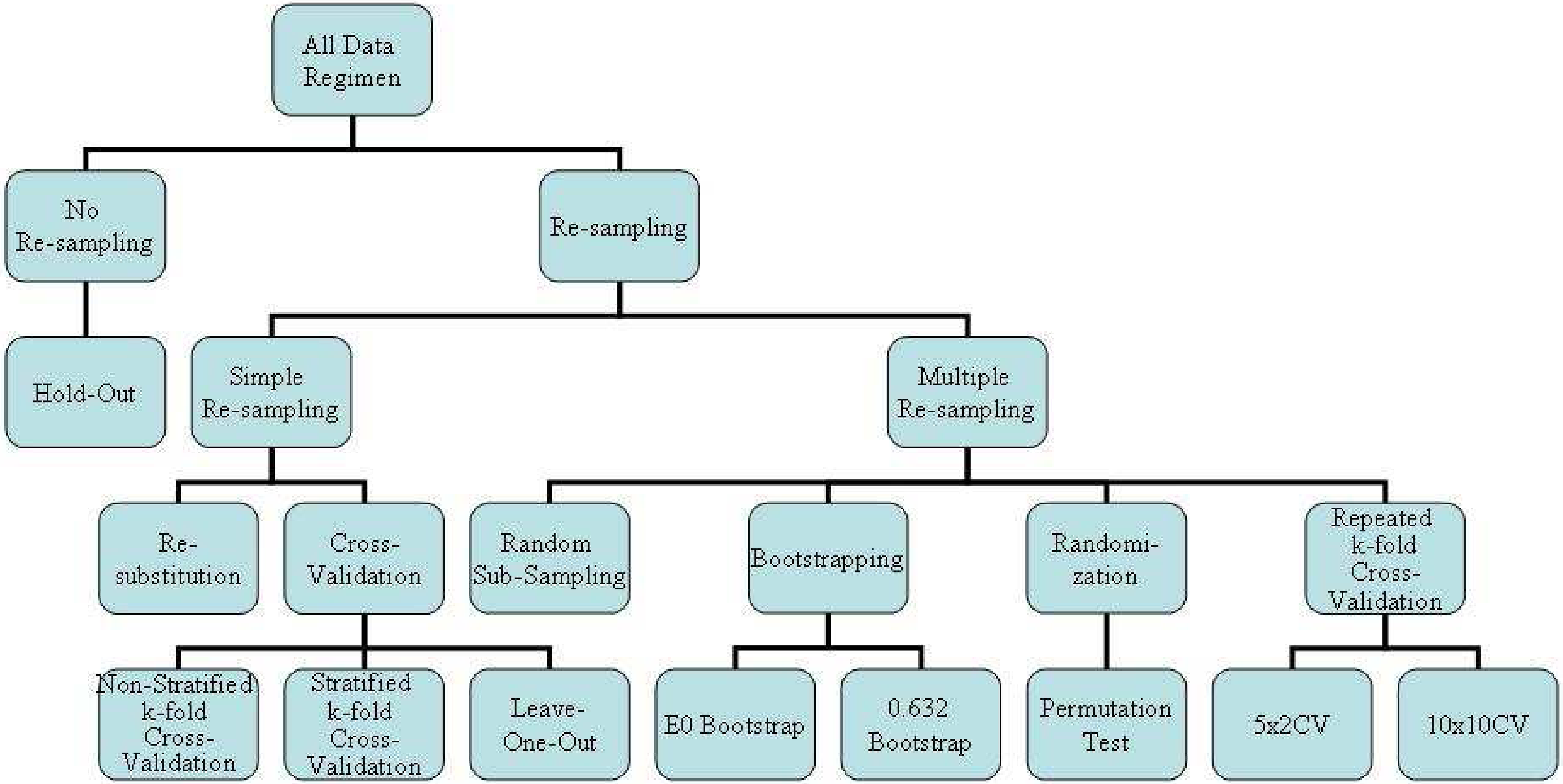
Internal

- Training k models can be expensive; want smaller k
- k-fold CV a reasonable choice because gives an almost unbiased estimate of accuracy

External

- Want to use hypothesis testing, e.g., Null Hypothesis is that the means of these two algorithms is the same
- Want a sampling technique that has less Type 1 errors
- Assumptions require independent samples of error, but empirically k-fold is not necessarily better than repeated sampling

Re-sampling summary



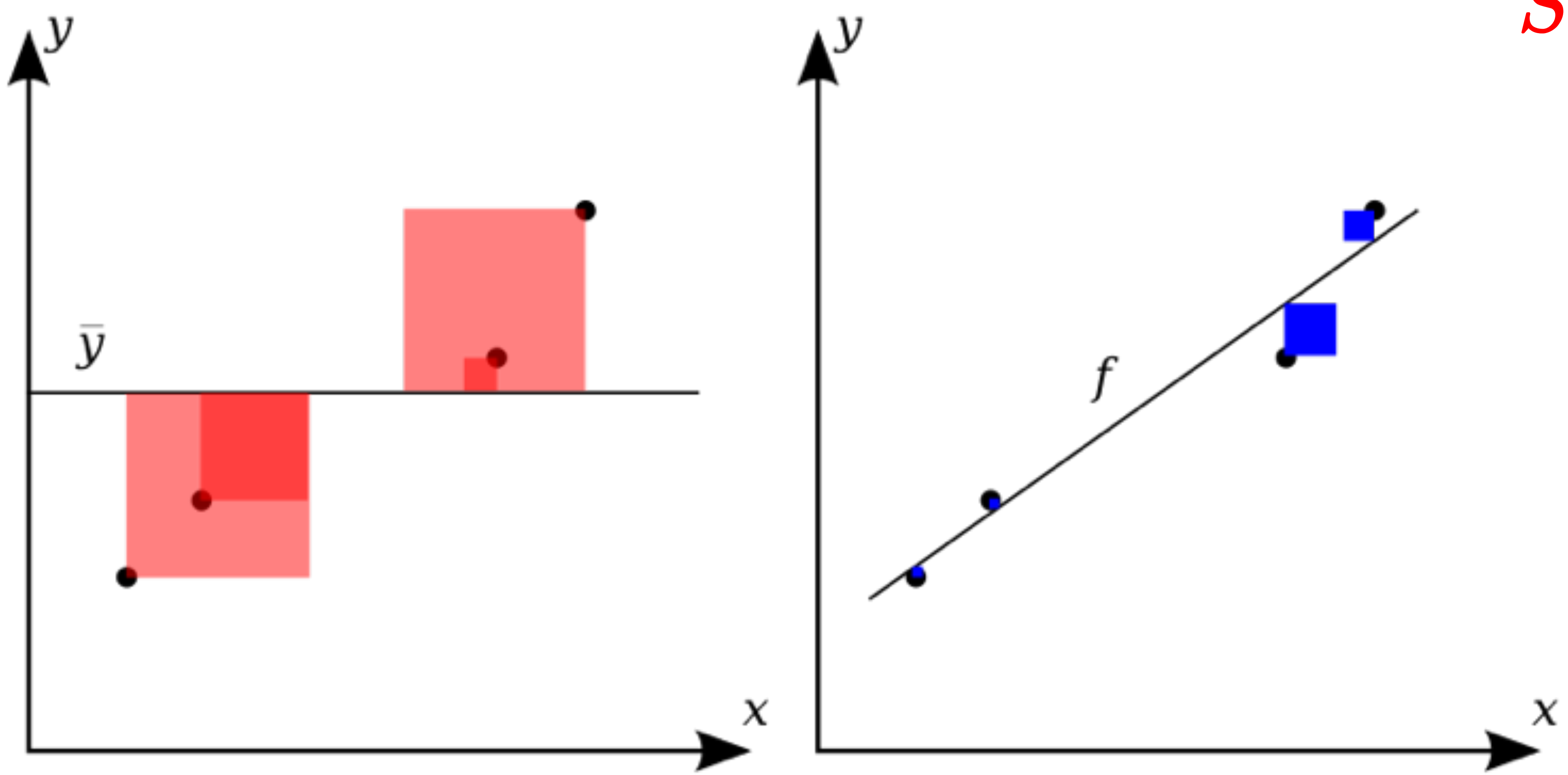
Regression objectives

- We have looked at l2 error for estimating parameters (i.e., as an objective) and to measure performance
- Other options:
  - l1 error — can be difficult to optimize, still a useful measure of error
  - smooth l1 — smooth and convex, easier to optimize, not usually used as a measure of error (unless reporting accuracy of optimizer)
  - R-squared — coefficient of determination
  - Variance unexplained
  - Percentage error — rescale by magnitude of values

R-squared measure

- Also called "coefficient of determination"

$$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}$$



- The sum of squares of residuals, also called the residual sum of squares:

$$SS_{\text{res}} = \sum_i (y_i - f_i)^2$$

- The total sum of squares (proportional to the variance of the data):

$$SS_{\text{tot}} = \sum_i (y_i - \bar{y})^2,$$

Larger R-squared is better

R-squared is monotone in number of features

- As add more features, the R-squared measure cannot decrease. Why?

$$R^2 \equiv 1 - \frac{SS_{\rm res}}{SS_{\rm tot}}. \qquad SS_{\rm res} = \sum_i (y_i - f_i)^2 \qquad SS_{\rm tot} = \sum_i (y_i - \bar{y})^2,$$

- Is this an issue?

- Alternative: adjusted R-squared — penalize the number of explanatory variables (features)

Percentage error

- If use error ll val1 - val2 ll, and get 0.1, is this good?
- One option: mean percentage error (issues?)
- Another option: mean absolute percentage error (MAPE)

$$\mathrm{M} = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|,$$

- Another option: symmetric MAPE

$$\mathrm{SMAPE} = \frac{1}{n}\sum_{t=1}^{n}\frac{|F_t - A_t|}{(|A_t| + |F_t|)/2}$$

Classification terminology

- True positives — samples predicted by classifier to be positive that have true label positive
- False positives — samples predicted by classifier to be positive that have true label negative
- True negatives — samples predicted by classifier to be negative that have true label negative
- False negatives — samples predicted by classifier to be negative that have true label positive

Classification measures

Name	Symbol	Definition
Classification error	$error$	$error = \frac{fp+fn}{tp+fp+tn+fn}$
Classification accuracy	$accuracy$	$accuracy = 1 - error$
True positive rate	tpr	$tpr = \frac{tp}{tp+fn}$
False negative rate	fnr	$fnr = \frac{fn}{tp+fn}$
True negative rate	tnr	$tnr = \frac{tn}{tn+fp}$
False positive rate	fpr	$fpr = \frac{fp}{tn+fp}$
Precision	pr	$pr = \frac{tp}{tp+fp}$
Recall	rc	$rc = \frac{tp}{tp+fn}$

Why these specific values?

- These measures exist for multiple reasons
- Separate the importance of false positives and false negatives
  - In some cases, much more hazardous to have a false positive than a false negative (or vice versa)
- Avoid issues with imbalanced datasets

Confusion Matrix for binary classification

True class

Predicted class

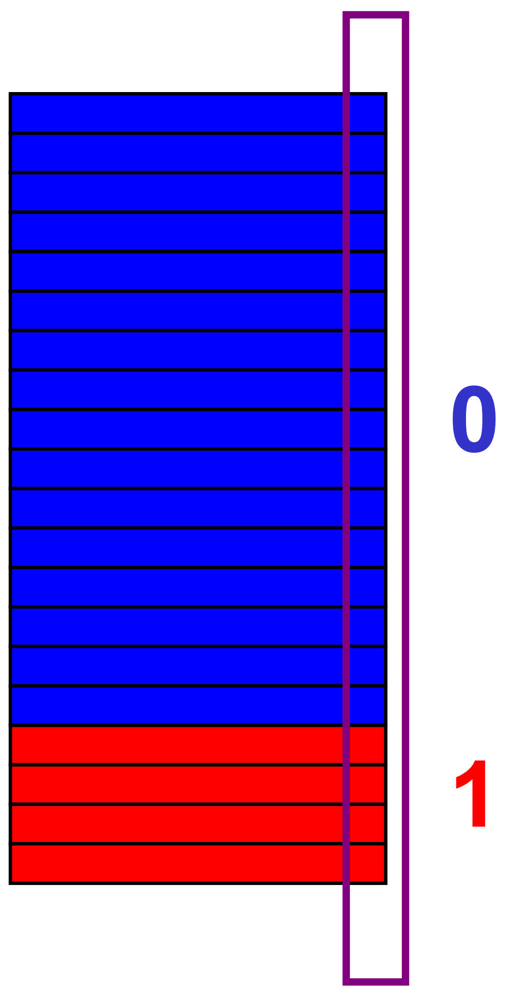
	0	1
0	N_{00}	N_{01}
1	N_{10}	N_{11}

Number of data points whose true class was 0 but predicted class was 1.

$$Accuracy = \frac{N_{00} + N_{11}}{N_{00} + N_{10} + N_{01} + N_{11}}$$

$$Error = 1 - Accuracy$$

Example of importance of measures: imbalanced datasets



16 data points have class 0 (majority class)

4 data points have class 1 (minority class)

Trivial classifier: always predict majority class

Accuracy of a trivial classifier is: 16/20 = 80%

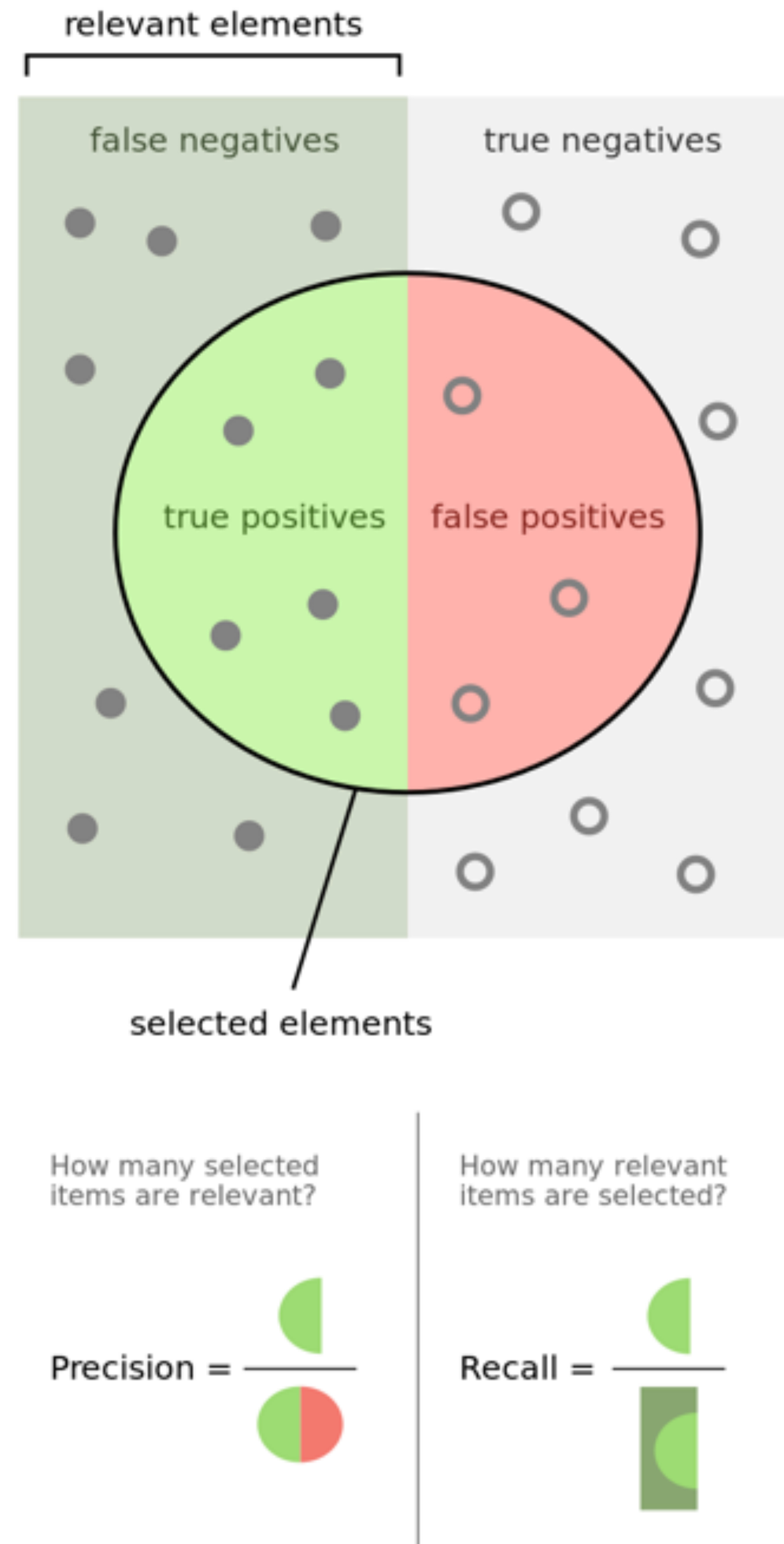
Random classifier: predict class 0 with probability 0.8 and class 1 with probability 0.2

Accuracy of the random classifier: 68%

($0.8^2 + 0.2^2 = 0.68$)

Precision and recall

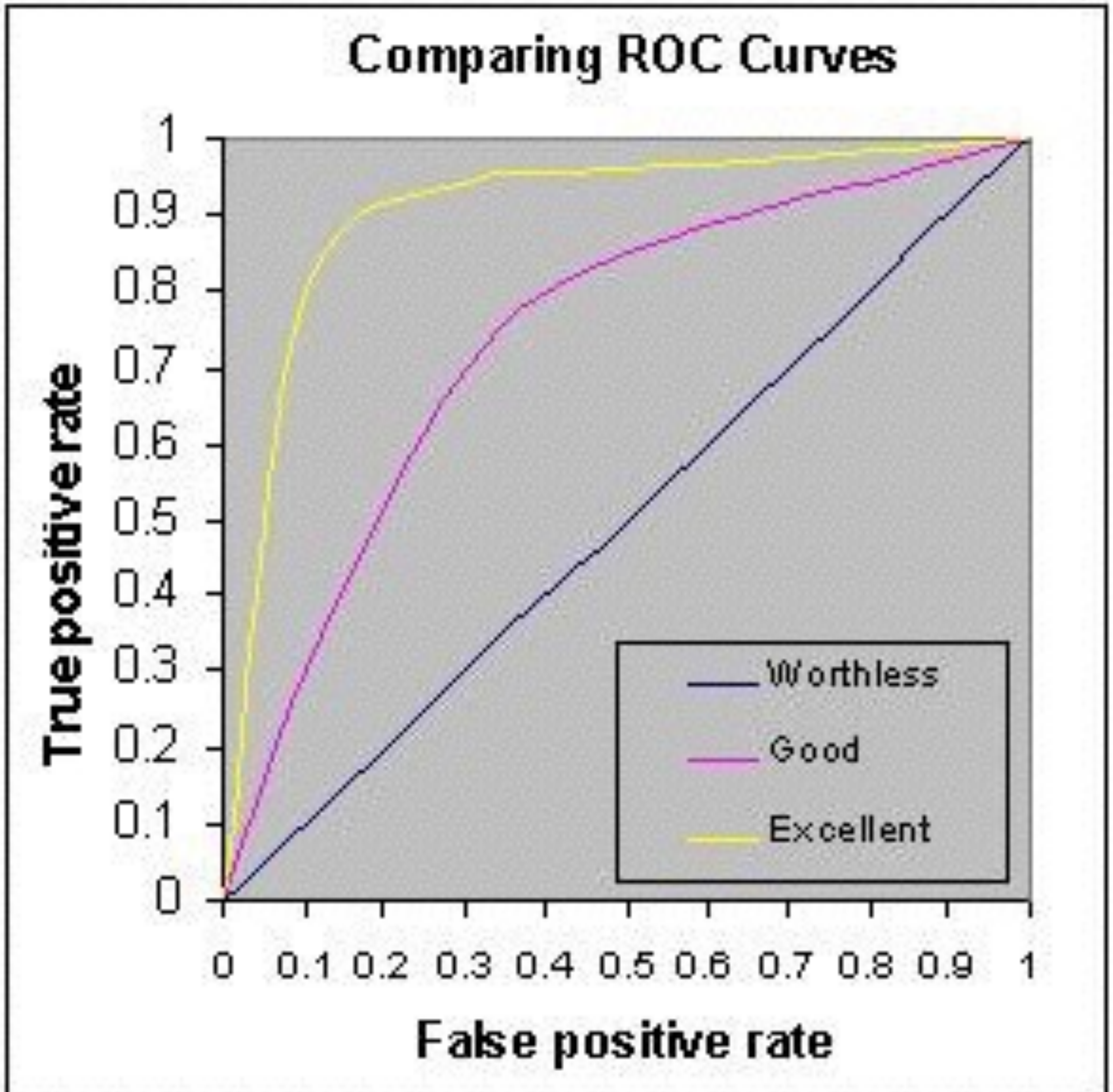
- Example: when a search engine returns 30 pages only 20 of which were relevant while failing to return 40 additional relevant pages, its precision is 20/30 = 2/3 while its recall is 20/60 = 1/3.



ROC Curve

Predict positive if
$p(y=1|x) >$ threshold

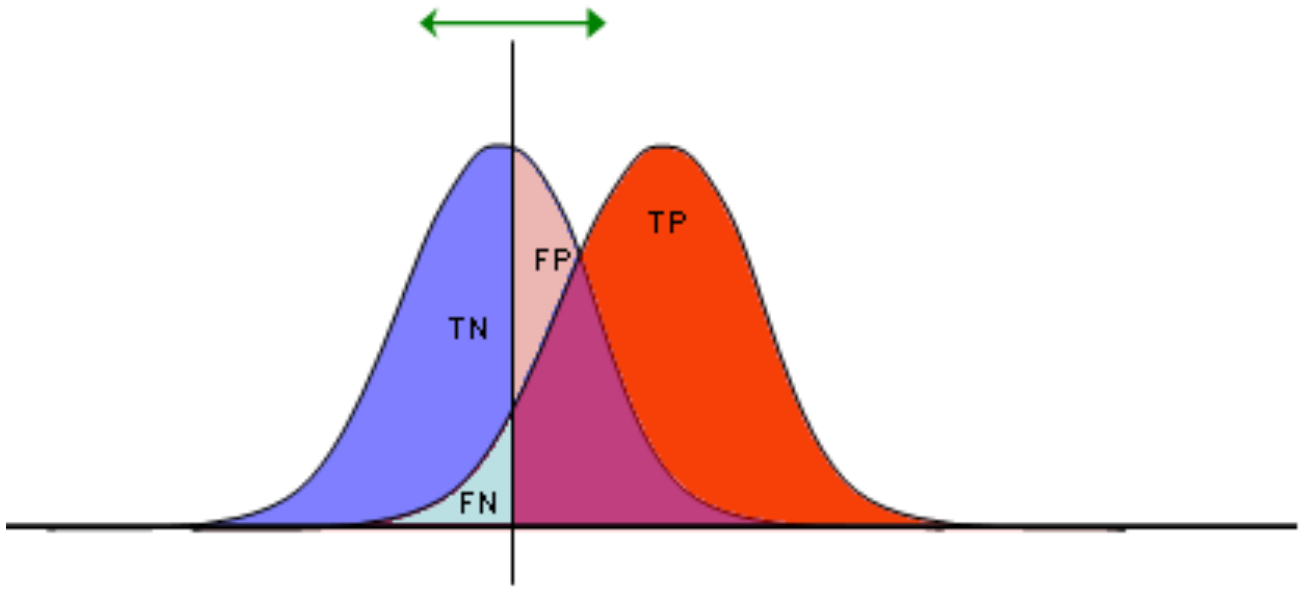
Threshold = 0



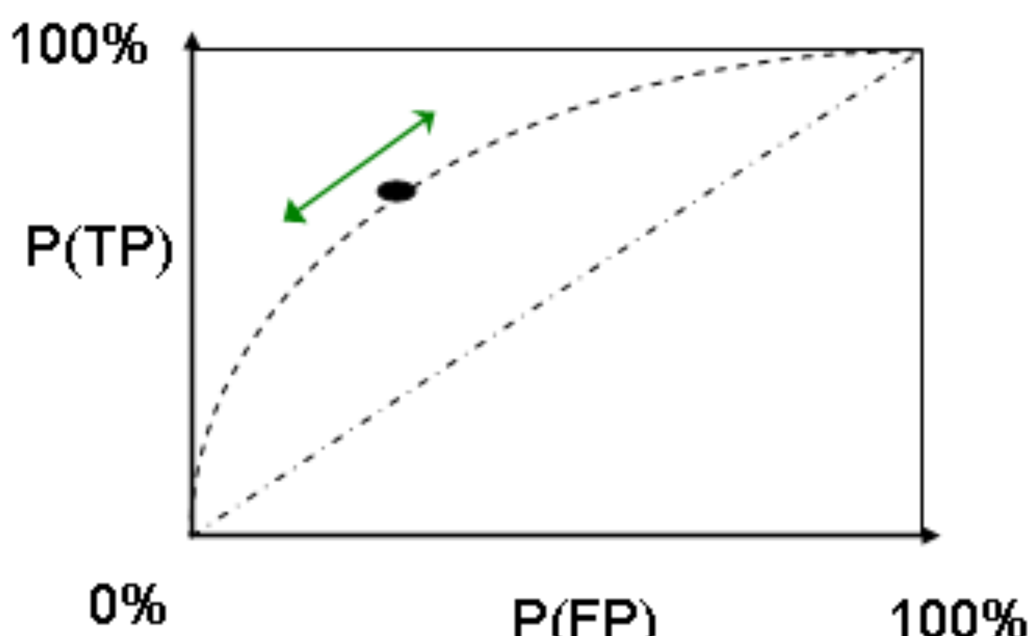
Threshold = 1

ROC Curve example

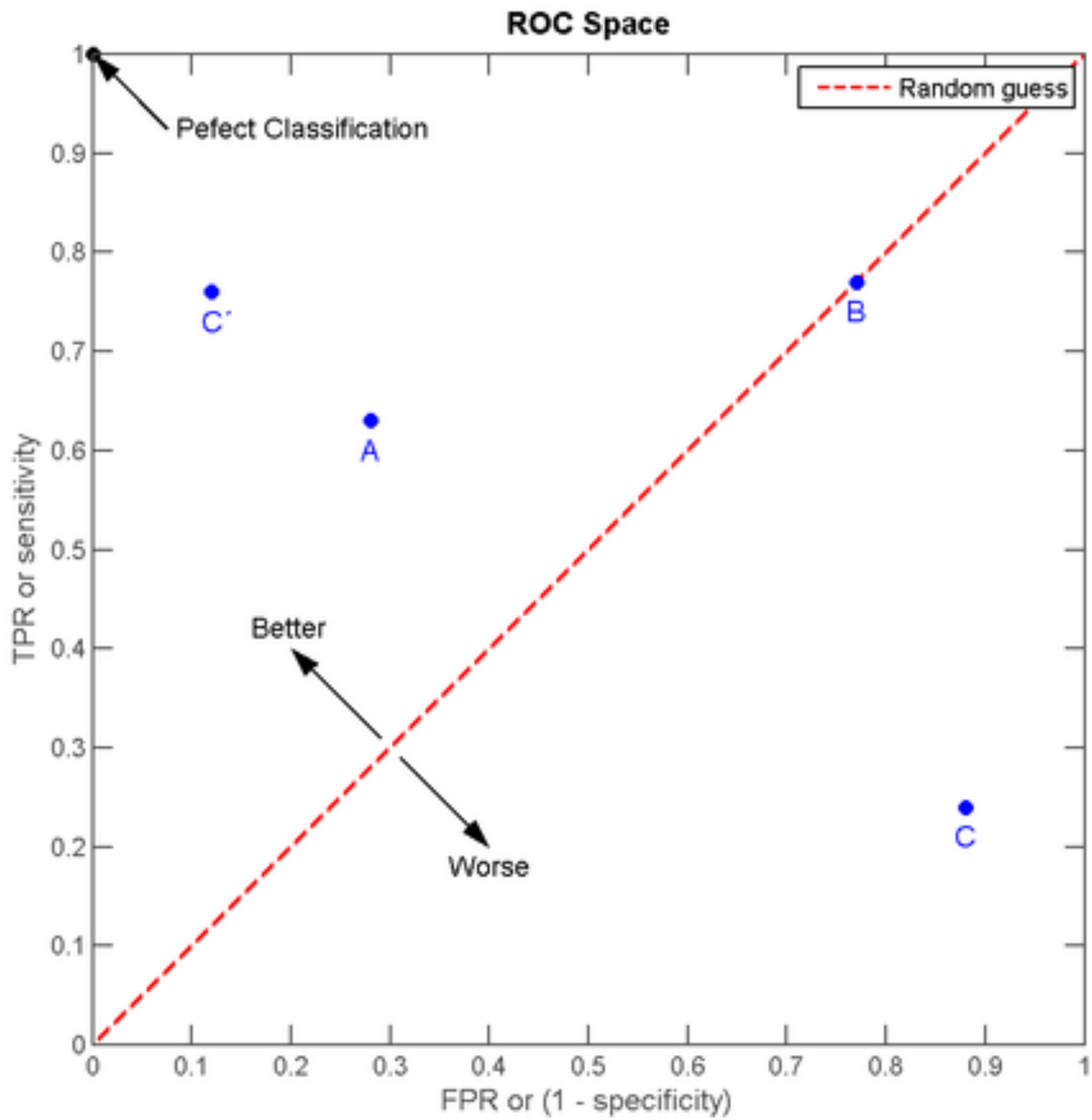
e.g., diseased people, healthy people
blood protein levels normally distributed
Parameter that changes: threshold



TP	FP
FN	TN
1	1

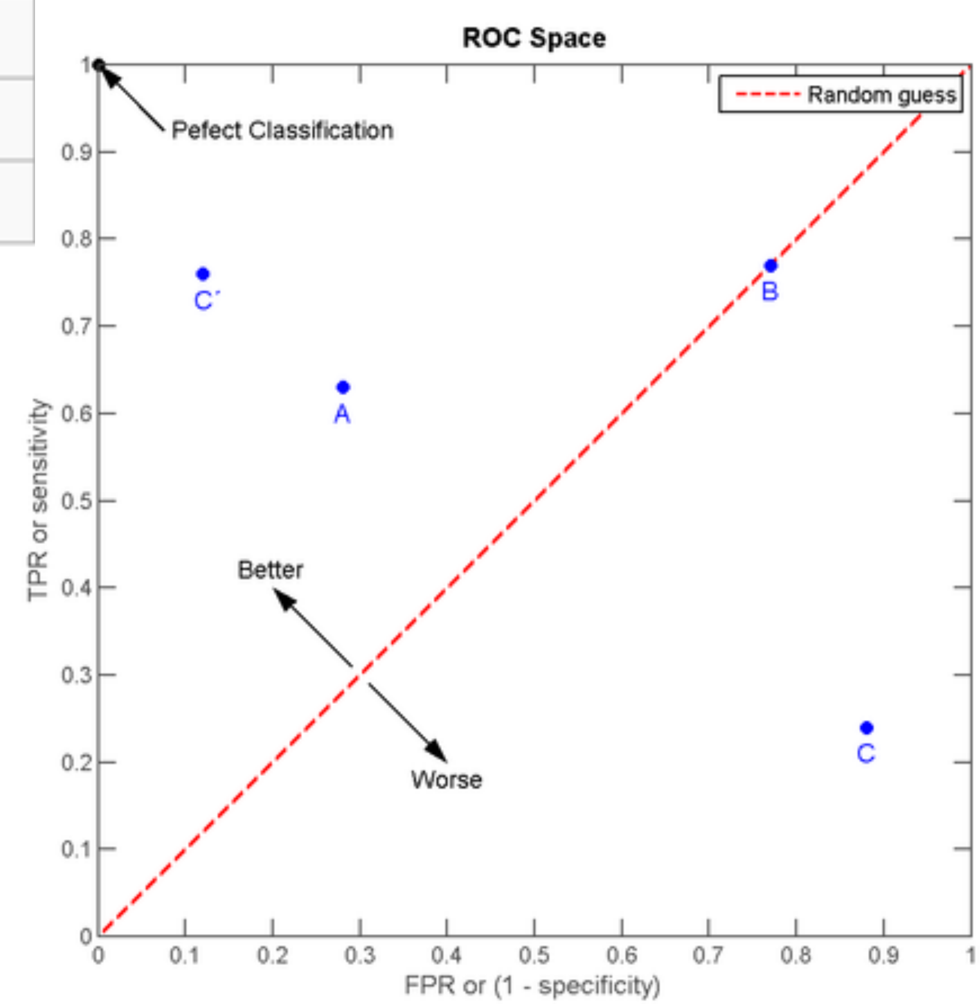


ROC space


ROC Space
Random guess
Pefect Classification
C'
B
A
Better
Worse
C
TPR or sensitivity
FPR or (1 - specificity)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

ROC space

A			B			C		
TP=63	FP=28	91	TP=77	FP=77	154	TP=24	FP=88	112
FN=37	TN=72	109	FN=23	TN=23	46	FN=76	TN=12	88
100	100	200	100	100	200	100	100	200
TPR = 0.63			TPR = 0.77			TPR = 0.24		
FPR = 0.28			FPR = 0.77			FPR = 0.88		
PPV = 0.69			PPV = 0.50			PPV = 0.21		
F1 = 0.66			F1 = 0.61			F1 = 0.22		
ACC = 0.68			ACC = 0.50			ACC = 0.18		



Area under the curve

- AUC or AUCROC gives the area under the ROC curve
- AUC is equal to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one
- Some issues in using AUC to compare classifiers (see "Measuring classifier performance: a coherent alternative to the area under the ROC curve", Hand, JMLR, 2009)
    - can give unequal important to a FPR or TPR for different classifiers