ROC Curves

Confusion Matrix

- The two class confusion matrix:

		Predicted class	
		Yes	No
Actual class	Yes	True positive (TP)	False negative (FN)
	No	False positive (FP)	True negative (TN)

True positives and False positives

- True positive rate is TPR = TP/P

  TPR = # of correctly classified as p /

  # of positives in the test data

- False positive rate is FPR = FP/N

  FPR = # of incorrectly classified as p /

  # of negatives in the test data

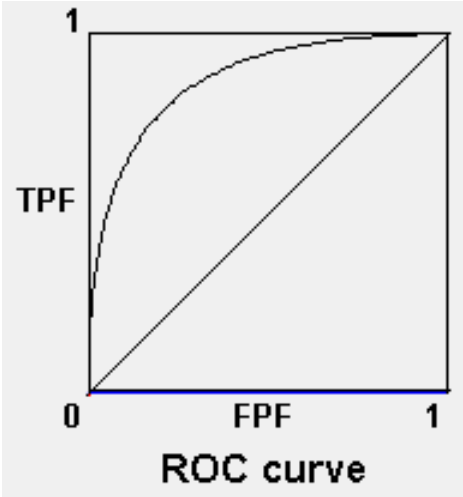
- Note that

  TP + FN = P

  TN + FP = N

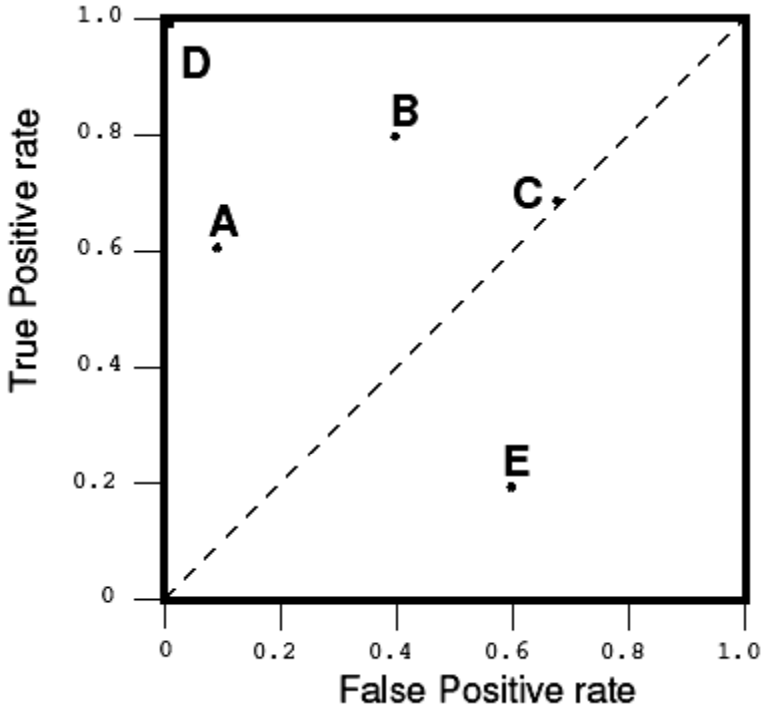
ROC

- ROC: Receiver Operating Characteristic
- It is a performance graphing method.
- A plot of True positive (TP) and false positive (FP) rates (fractions).
- Used for evaluating data mining schemes, and comparing the relative performance among different classifiers.



ROC Space

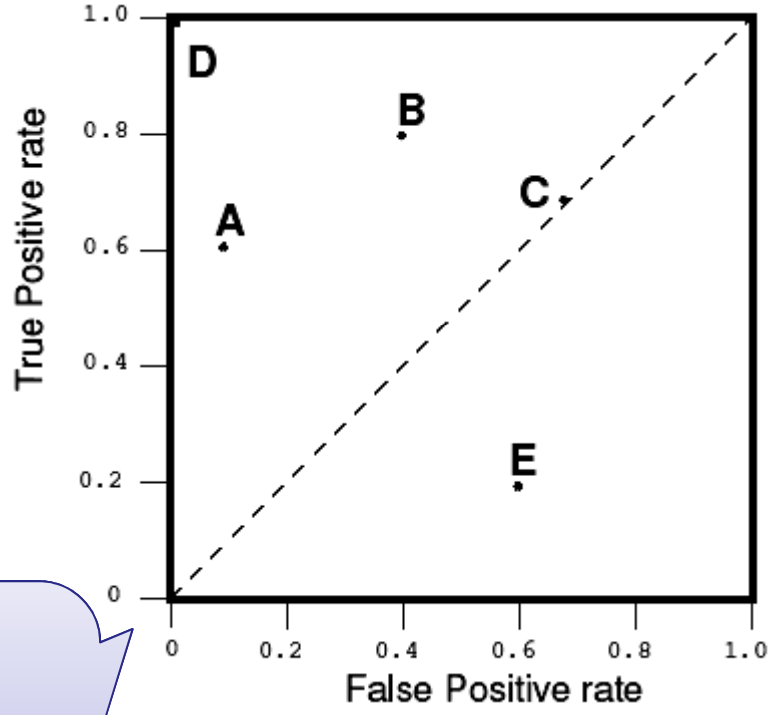
- ROC space:
  - TPR is plotted on the Y axis and
  - FPR is plotted on the X axis.
  - depicts relative trade-offs between
    - **benefits** (true positives) and
    - **costs** (false positives).



- Figure shows a ROC graph with five discrete classifiers labeled A through E.
- Each discrete classifier has one (fpr, tpr) pair corresponding to a single point in ROC space.

A discrete classifier is one that outputs only a class label.

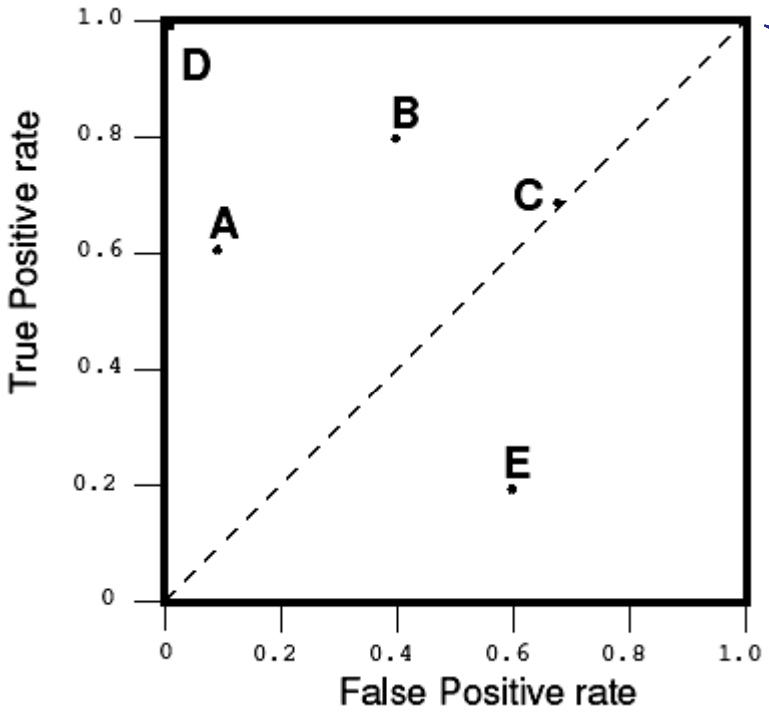
Lower left point (0, 0)



Never issue a positive classification!

such a classifier commits
no false positive errors
but also gains
no true positives.

Upper Right corner (1, 1)



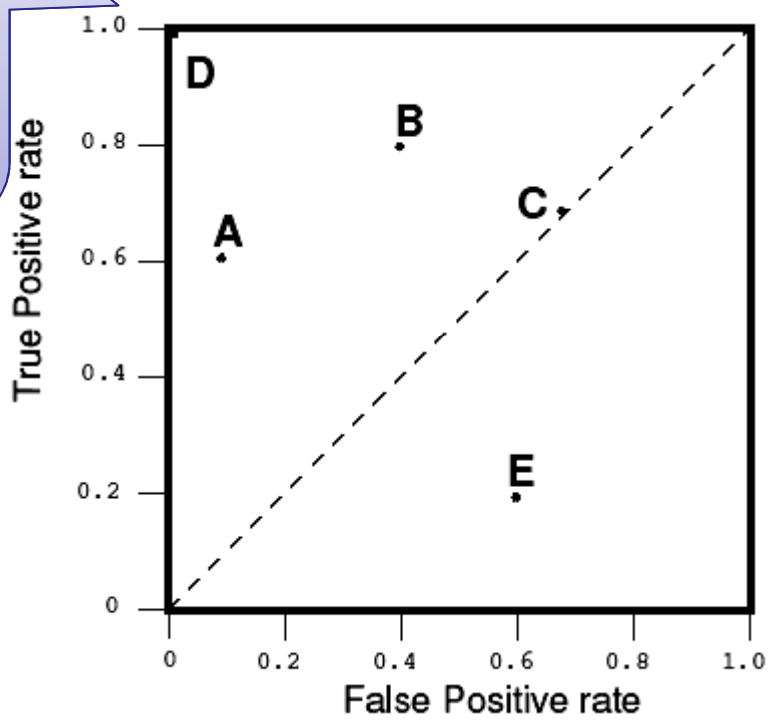
Unconditionally issue positive classification!

such a classier predicts
all positive instances correctly
but at the cost of predicting
all negative instances wrongly

Point D (0,1)

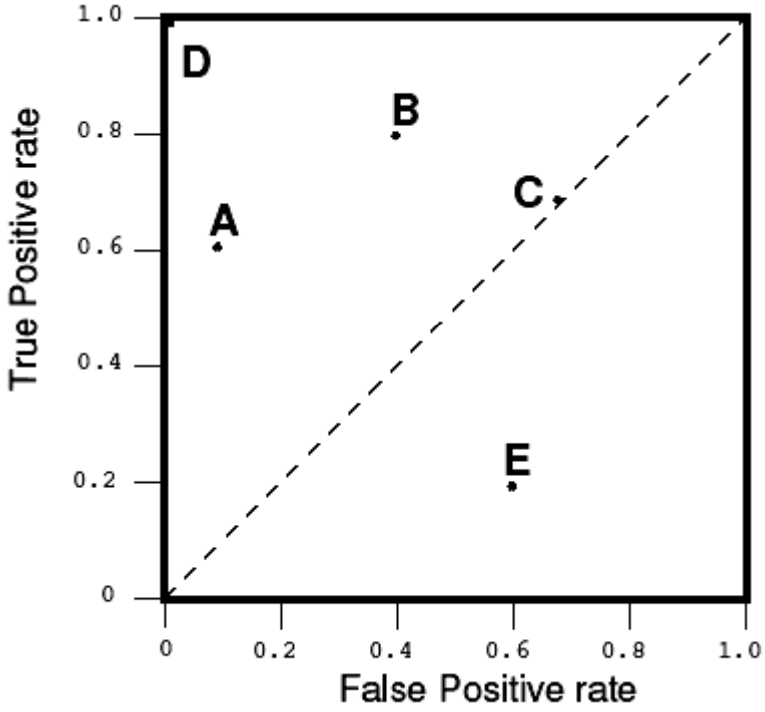
Get everything perfect!

this perfect classier commits
no false positive errors
and gets
all true positives



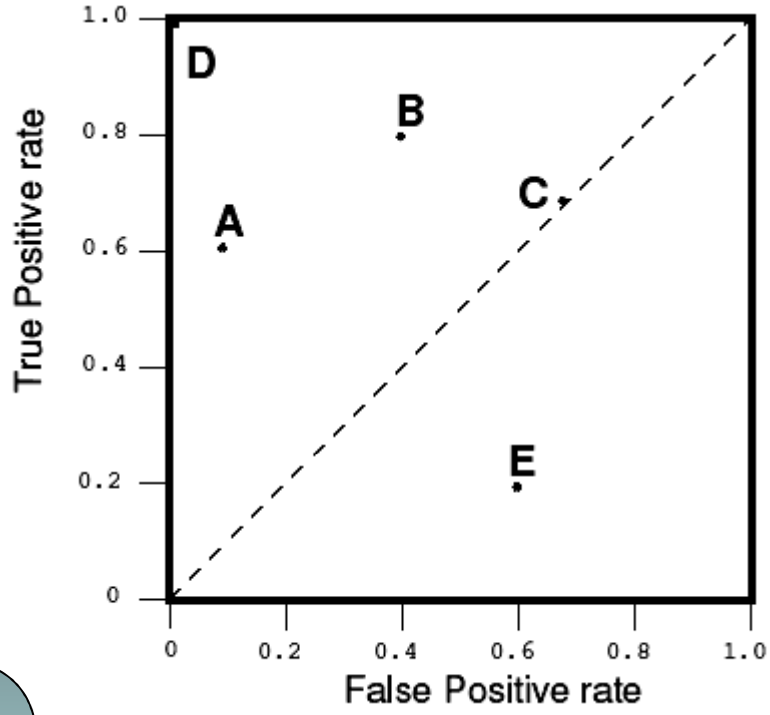
Several Points in ROC Space

- A point in ROC space is better than another if it is to the **northwest** of the other, i.e.
  - TP rate is higher,
  - FP rate is lower, or both.



Point C: Random Performance

- The diagonal line y = x represents the strategy of randomly guessing a class.

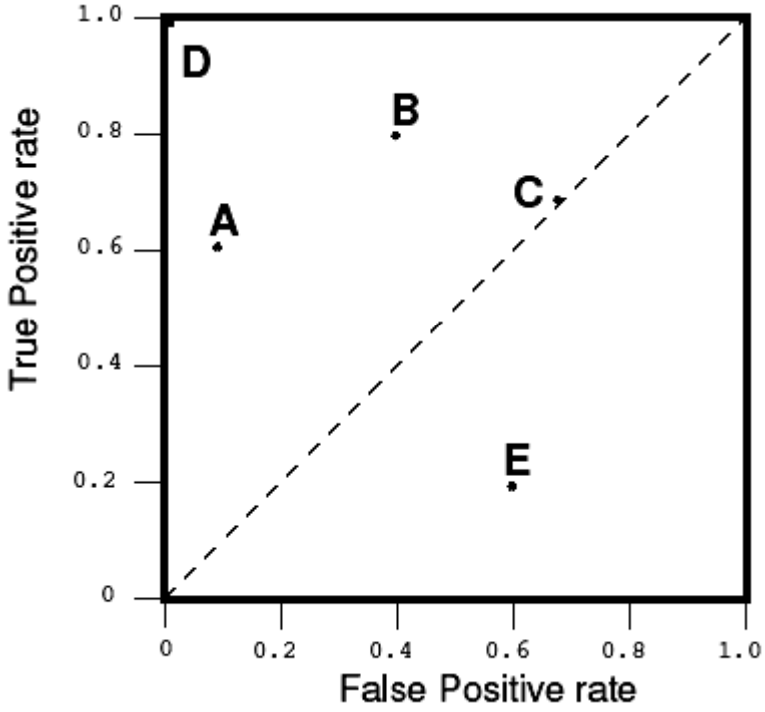


C's performance is virtually random.

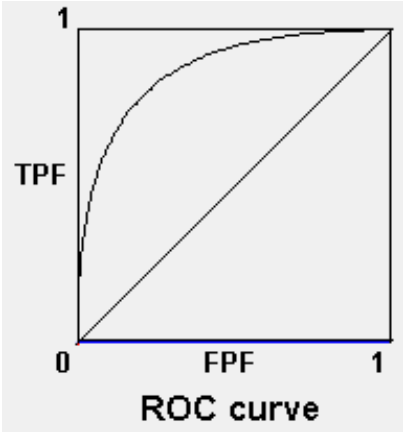
At (0.7, 0.7), C is guessing the positive class 70% of the time.

Upper and Lower Triangular Areas

- Any classifier that appears in the lower right triangle performs worse than random guessing.
  - This triangle is therefore usually empty in ROC graphs.



Curves in ROC space



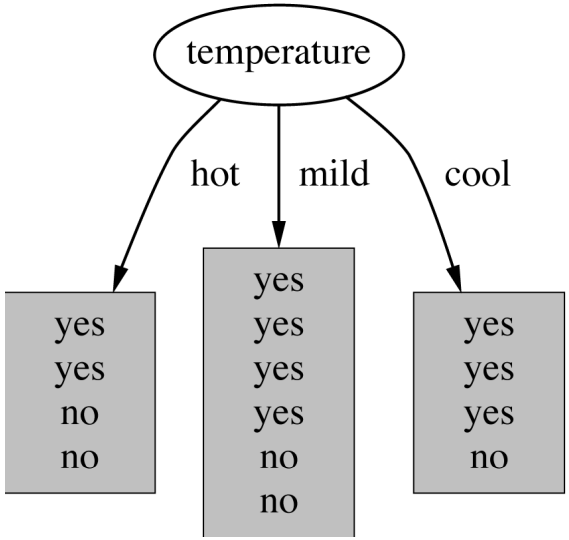
- Many classifiers, such as decision trees or rule sets, are designed to produce only a class decision, i.e., a Y or N on each instance.
  - When such a discrete classier is applied to a test set, it yields a single confusion matrix, which in turn corresponds to one ROC point.

- Some classifiers, such as a Naive Bayes classifier, yield an instance probability or score.
  - Such a ranking or scoring classier can be used with a threshold to produce a discrete (binary) classier:
    - if the classier output is above the threshold, the classier produces a Y,
    - else it produces an N.
  - Each different threshold value produces a different point in ROC space (corresponding to a different confusion matrix).

Creating Scoring Classifiers

- Many discrete classifier models may easily be converted to scoring classifiers by "looking inside" them at the instance statistics they keep.

- For example, a decision tree determines a class label of a leaf node from the proportion of instances at the node; the class decision is simply the most prevalent class.
  - These class proportions may serve as a score.



Algorithm

- Exploit monotonicity of thresholded classifications:
  - Any instance that is classified positive with respect to a given threshold, will be classified positive for all lower thresholds as well.

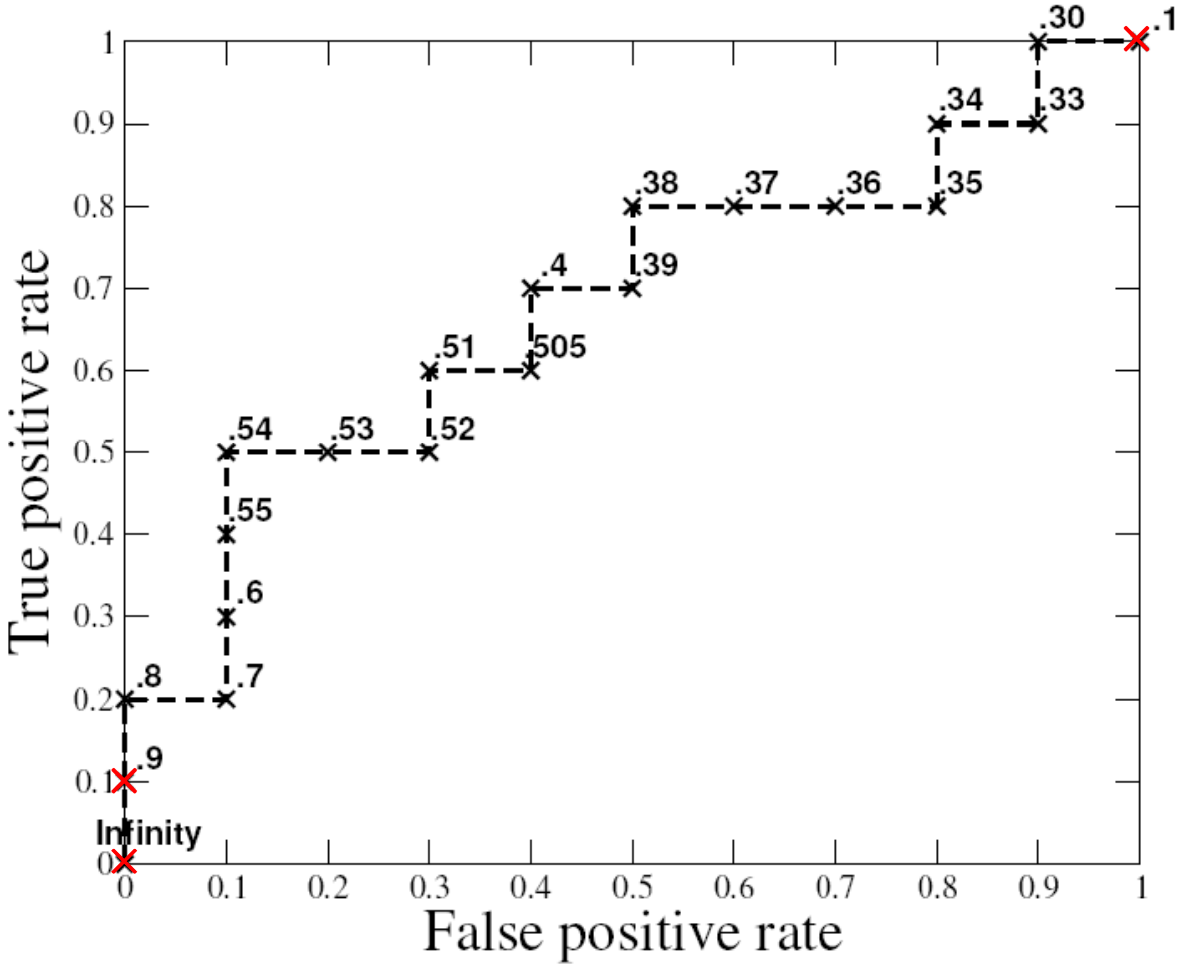
- Therefore, we can simply:
  - sort the test instances decreasing by their scores,
  - move down the list (lowering the threshold), processing one instance at a time, and
  - update TPR and FPR as we go.

- In this way, a ROC graph can be created from a linear scan.

Example

Inst#	Class	Score	Inst#	Class	Score
1	**p**	.9	11	**p**	.4
2	**p**	.8	12	**n**	.39
3	**n**	.7	13	**p**	.38
4	**p**	.6	14	**n**	.37
5	**p**	.55	15	**n**	.36
6	**p**	.54	16	**n**	.35
7	**n**	.53	17	**p**	.34
8	**n**	.52	18	**n**	.33
9	**p**	.51	19	**p**	.30
10	**n**	.505	20	**n**	.1

Example



A threshold of +inf produces the point (0, 0).

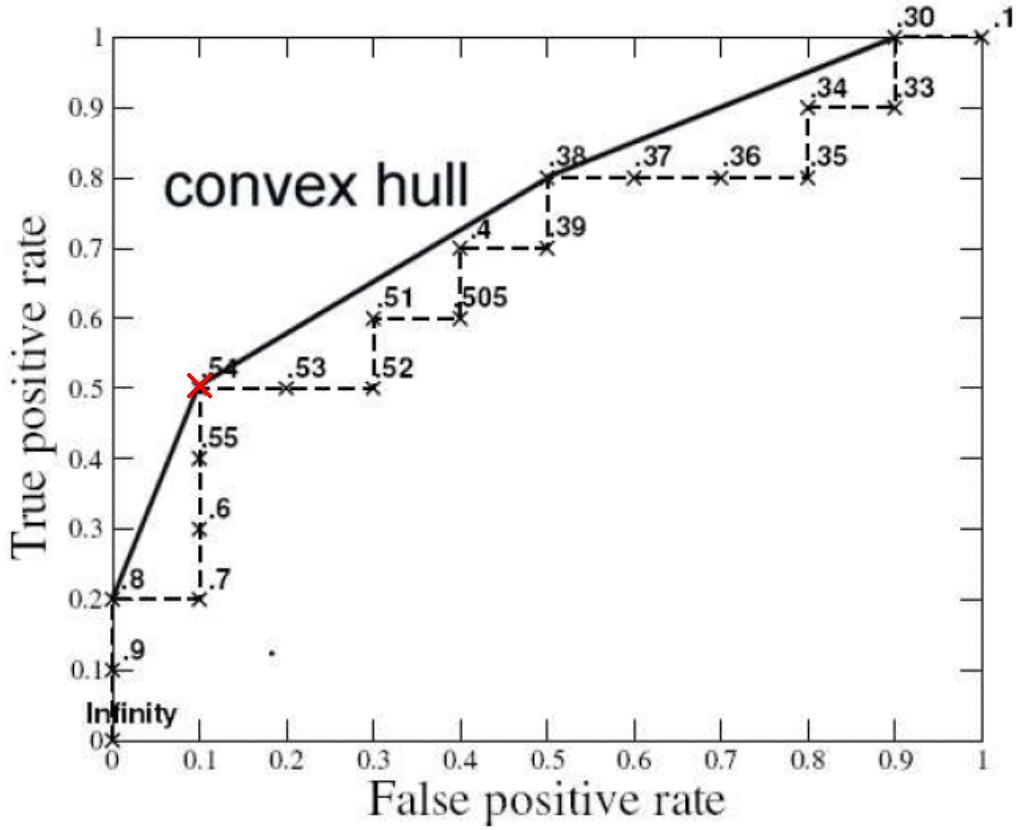
As we lower the threshold to 0.9 the first positive instance is classified positive, yielding (0, 0.1).

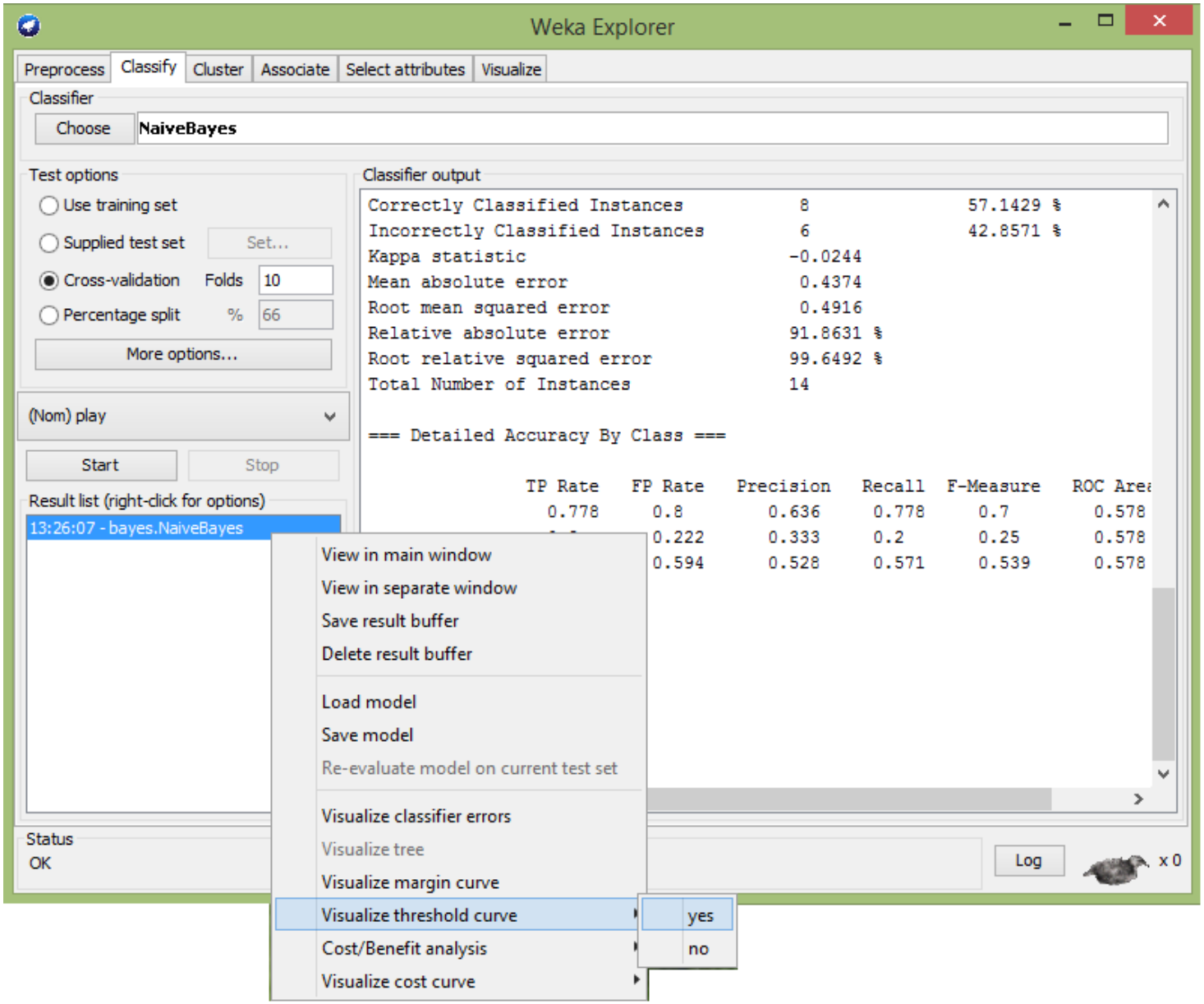
As the threshold is further reduced, the curve climbs up and to the right, ending up at (1, 1) with a threshold of 0.1.

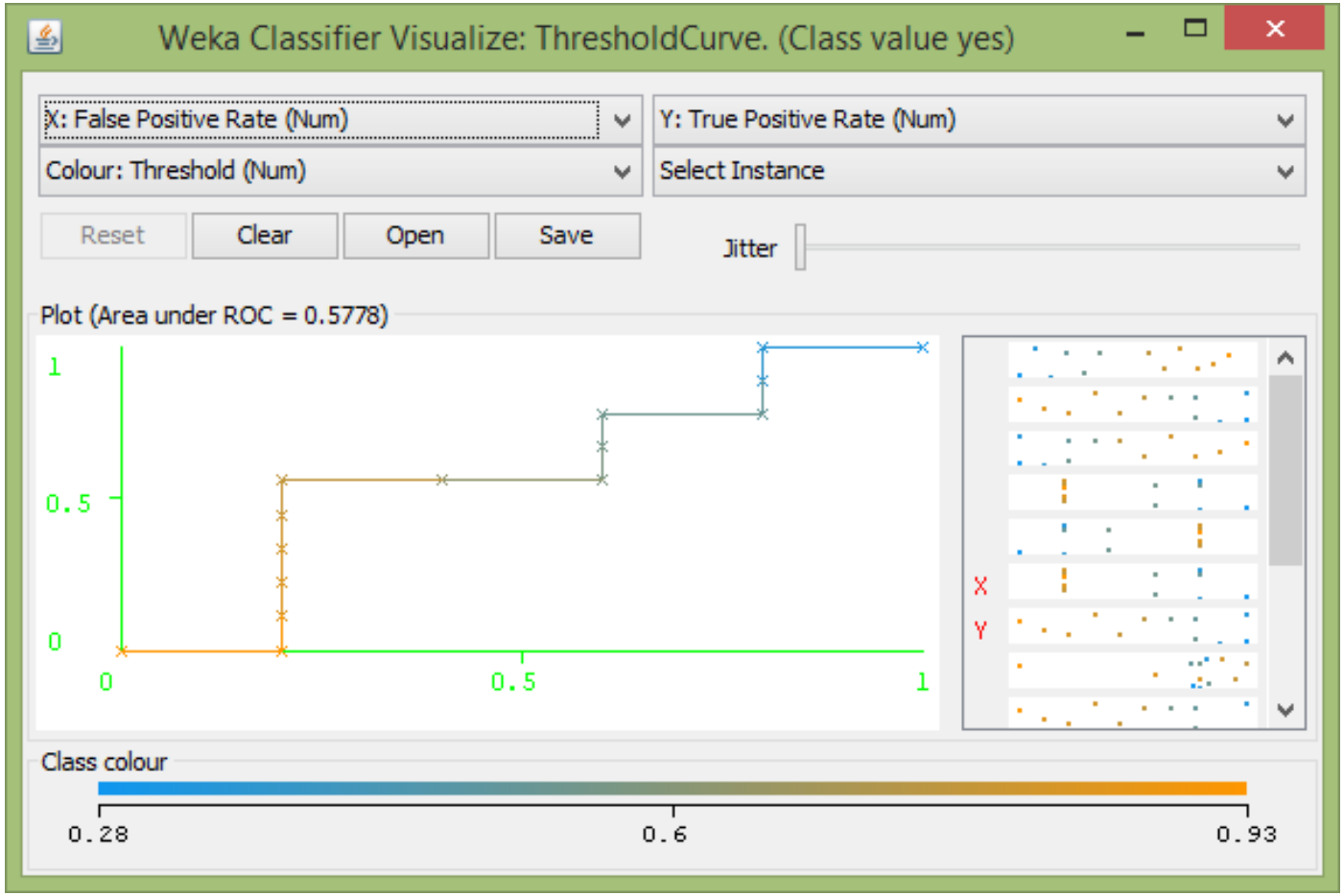
Observations – Accuracy

- The ROC point at (0.1, 0.5) produces its highest accuracy (70%).

- Note that the classifier's best accuracy occurs at a threshold of .54, rather than at .5 as we might expect with a balanced class distribution.

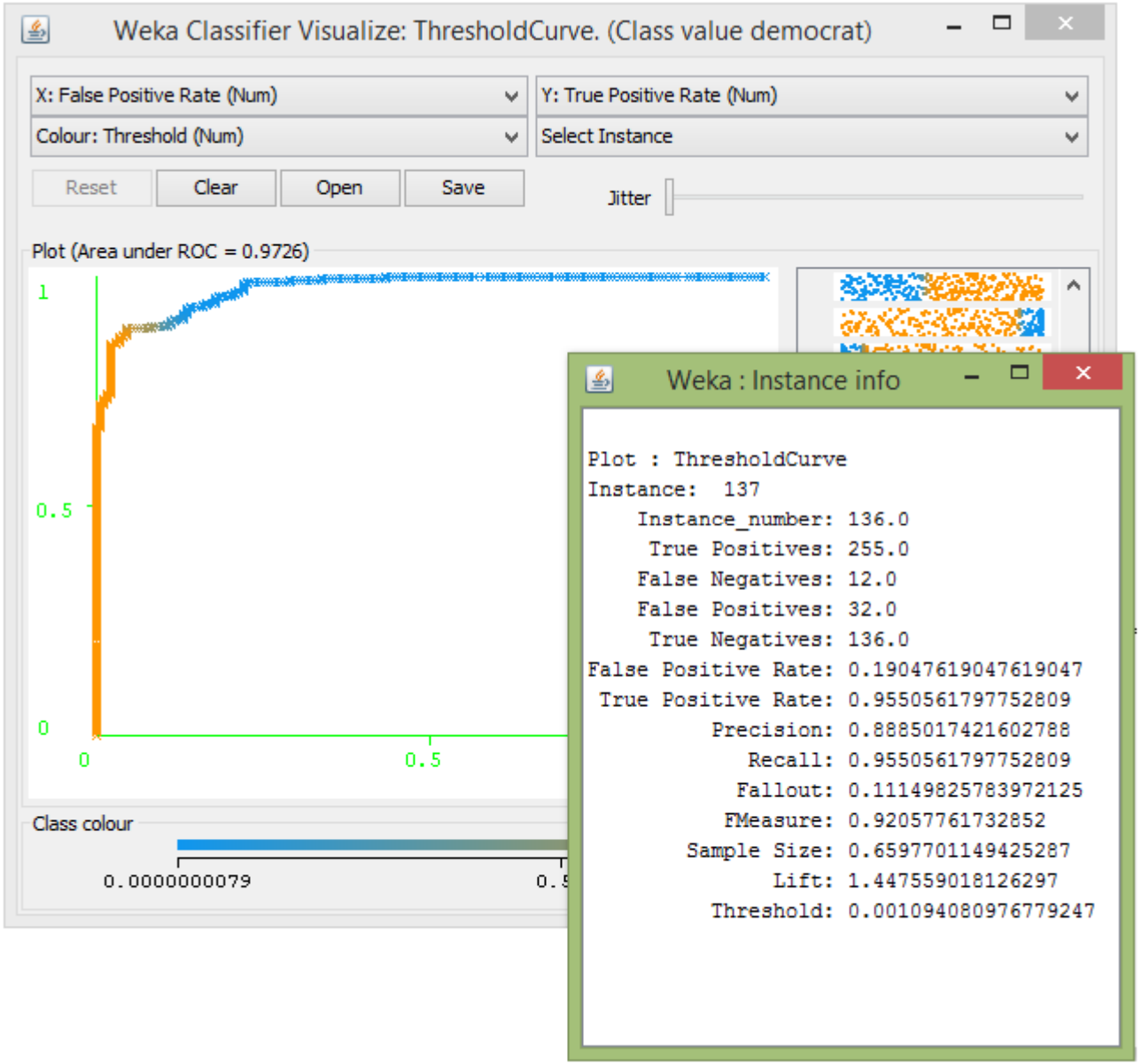



Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
NaiveBayes
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
Classifier output
Correctly Classified Instances 8 57.1429 %
Incorrectly Classified Instances 6 42.8571 %
Kappa statistic -0.0244
Mean absolute error 0.4374
Root mean squared error 0.4916
Relative absolute error 91.8631 %
Root relative squared error 99.6492 %
Total Number of Instances 14
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure ROC Area
0.778 0.8 0.636 0.778 0.7 0.578
0.222 0.333 0.2 0.25 0.578
0.594 0.528 0.571 0.539 0.578
(Nom) play
Start
Stop
Result list (right-click for options)
13:26:07 - bayes.NaiveBayes
View in main window
View in separate window
Save result buffer
Delete result buffer
Load model
Save model
Re-evaluate model on current test set
Visualize classifier errors
Visualize tree
Visualize margin curve
Visualize threshold curve
Cost/Benefit analysis
Visualize cost curve
yes
no
Status
OK
Log
x 0



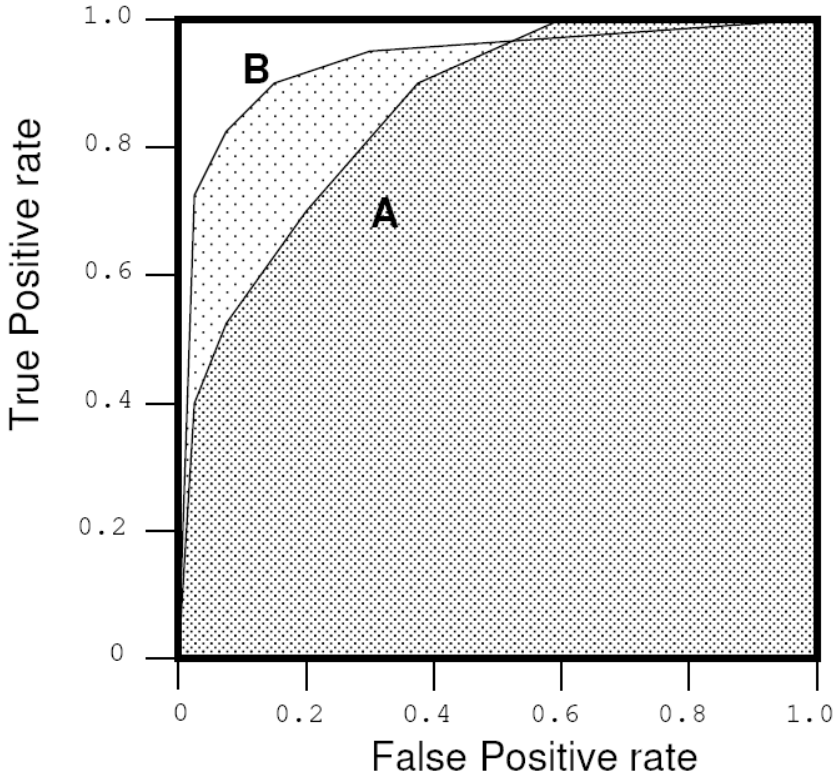
- Weka:Weather nominal dataset

- Weka:
  Vote dataset



Area under a ROC Curve

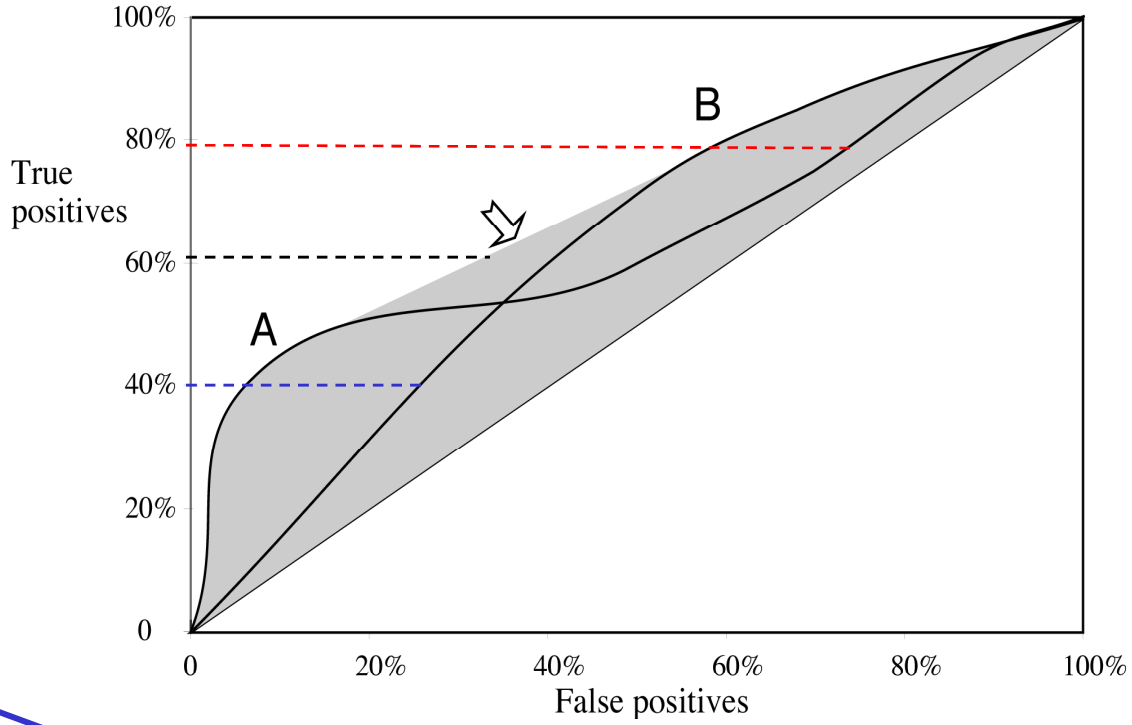
- Often used to compare classifiers:
  - The bigger AUC the better
- AUC can be computed by a slight modification to the algorithm for constructing ROC curves.



Convex Hull

- The shaded area is called the convex hull of the two curves.
- You should operate always at a point that lies on the upper boundary of the convex hull. E.g.
- What about some point in the middle* where neither A nor B lies on the convex hull?
- Answer:

  "Randomly" combine A and B



If you aim to cover just 40% of the true positives you should choose method A, which gives a btter false positive rate of 5%.

If you aim to cover 80% of the true positives you should choose method B, which gives a better false positive rate of 60% as compared with A's 80%.

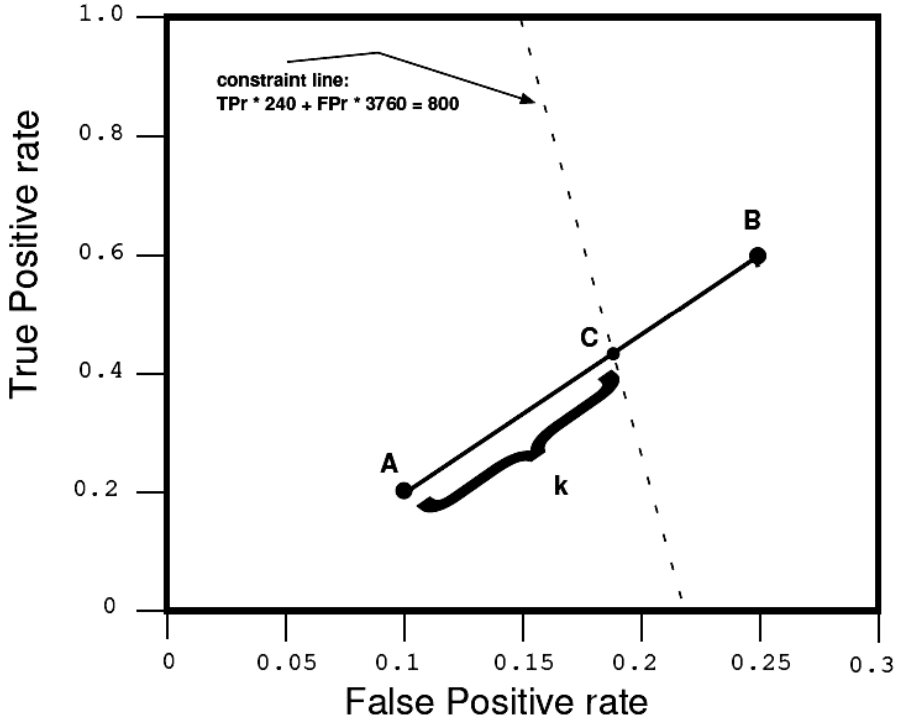
If you aim to cover 60% of the true positives then you should combine A and B.

Combining classifiers

- Example (CoIL Symposium Challenge 2000):
  - There is a set of 4000 clients to whom we wish to market a new insurance policy.
  - Our budget dictates that we can afford to market to only 800 of them, so we want to select the 800 who are most likely to respond to the offer.
  - The expected class of responders is 6%, so within the population of 4000 we expect to have 240 responders (positives) and 3760 non-responders (negatives).

Combining classifiers

- Assume we have generated two classifiers, A and B, which score clients by the probability they will buy the policy.
- In ROC space,
  - A's best point lies at (.1, .2) and
  - B's best point lies at (.25, .6)
- We want to market to exactly 800 people so our solution constraint is:
  - FPR * 3760 + TPR * 240 = 800
- If we use A, we expect:
  - .1 * 3760 + .2*240 = 424 candidates, which is too few.
- If we use B we expect:
  - .25*3760 + .6*240 = 1084 candidates, which is too many.



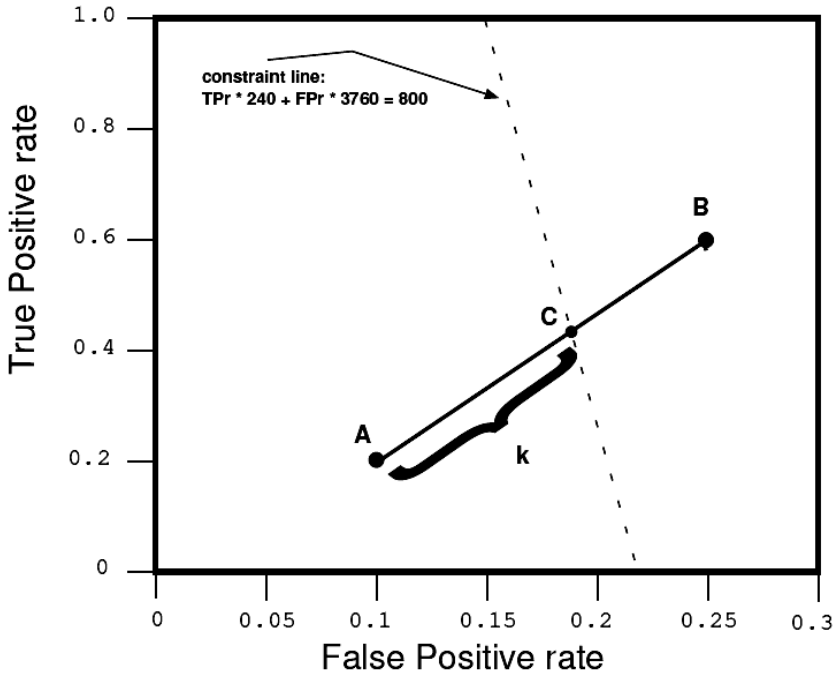
- We want a classifier between A and B.

Combining classifiers

- The solution constraint equation is shown as a dashed line.
- It intersects the line between A and B at C,
  - approximately (.18, .42)
- A classifier at point C would give the performance we desire and we can achieve it using linear interpolation.
- Calculate *k* as the ***proportional distance*** that C lies on the line between A and B:

  $k = (.18 - .1) / (.25 - .1) \approx 0.53$

- Therefore, if we sample B's decisions at a rate of .53 and A's decisions at a rate of 1-.53=.47 we should attain C's performance.



In practice this fractional sampling can be done as follows:

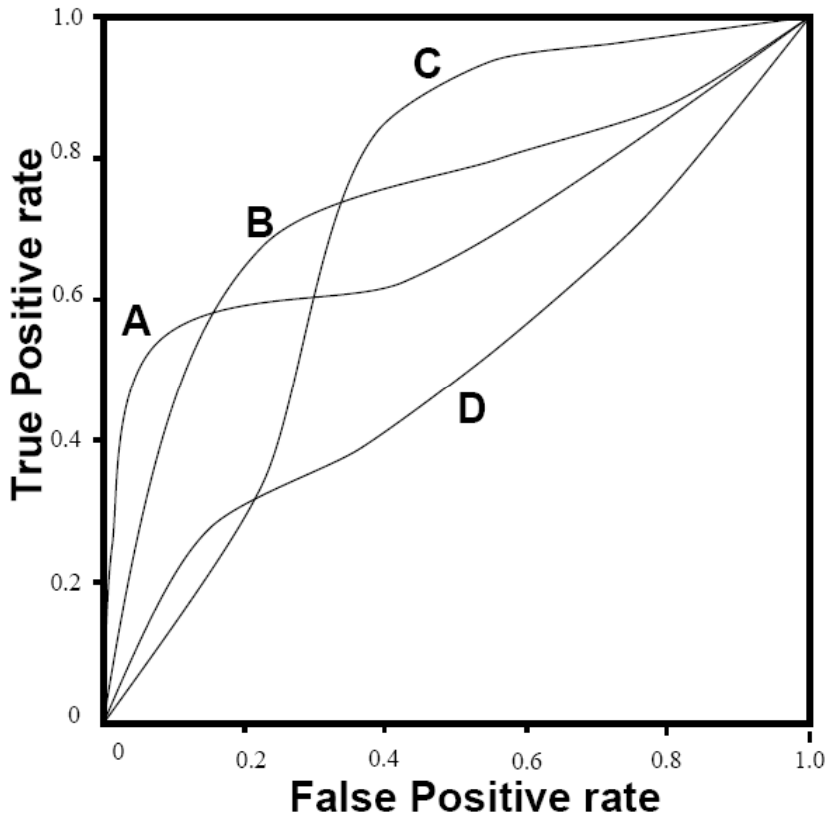
- For each instance (person), generate a random number between zero and one.
- If the random number is greater than *k*, apply classifier A to the instance and report its decision, otherwise pass the instance to B.

Convex Hull – Comparing Classifiers

- Classifier B is not useful if we have classifier A and classifier C.
  - Because B is all contained in the convex hull.

    So for any operating point of B, we can find a point *p* in the convex hull that is more northwest, i.e. better.

  - *p* can be obtained by combining A and C as described previously.



Measuring Classifier Performance

- **Accuracy** = (TP+TN) / (P+N) (same as *success rate*)
- **Precision** = TP / (TP+FP)
- **Recall** = TP / P (same as TPR, also called s*ensitivity* and *hit rate*)
- **F-measure** is the harmonic mean of the precision and recall:
  - F-measure = 2*Precision*Recall / (Precision + Recall)
- No single measure tells the whole story
- Use of multiple measures recommended
- Weka also produces **Fallout** = FP / (TP+FP)

Sensitivity and Specificity

- Popular in the medical domain

- **Sensitivity =** TP / P (same as TPR)
  - Example: Fraction of patients with cancer that the classifier is rightly predict.

- **Specificity** = TN / N
  - Example: Fraction of patients without cancer that the classifier rightly rules out.

- Often doctors like to increase sensitivity at the expense of specificity.

  Typically, achieved by lowering the threshold for predicting “Yes”.

The Inadequacy of Accuracy

- As the class distribution becomes more skewed, evaluation based on accuracy breaks down.
  - Consider a domain where the classes appear in a 999:1 ratio.
  - A simple rule, which classifies as the maximum likelihood class, gives a 99.9% accuracy.
  - Presumably this is not satisfactory if a non-trivial solution is sought.

- Evaluation by classification accuracy also tacitly assumes equal error costs, that is, a false positive error is equivalent to a false negative error.
  - In the real world this is rarely the case, because classifications lead to actions which have consequences, sometimes grave.

Counting the cost

- The two class *cost matrix*

		Predicted class	
		Yes	No
Actual class	Yes	0	1
	No	1	0

- Taking cost matrix into account replaces the success rate by *average cost per decision*.

Cost based classification

- Let {y, n} be the positive and negative class instances.
- Let {Y, N} be the classifications produced by a classifier.
- Let c(Y, n) be the cost of a false positive error.
- Let c(N, y) be the cost of a false negative error.

- For an instance E,
  - the classifier computes $p(y|E)$ and $p(n|E)=1- p(y|E)$ and
  - the decision to emit a positive classification is when

$$p(n|E)*c(Y, n) < p(y|E) * c(N, y)$$

i.e.

$$p(n|E)*c(Y, n) / c(N, y) < p(y|E)$$

Cost-based Classification

MetaCost (Domingos, 1999)

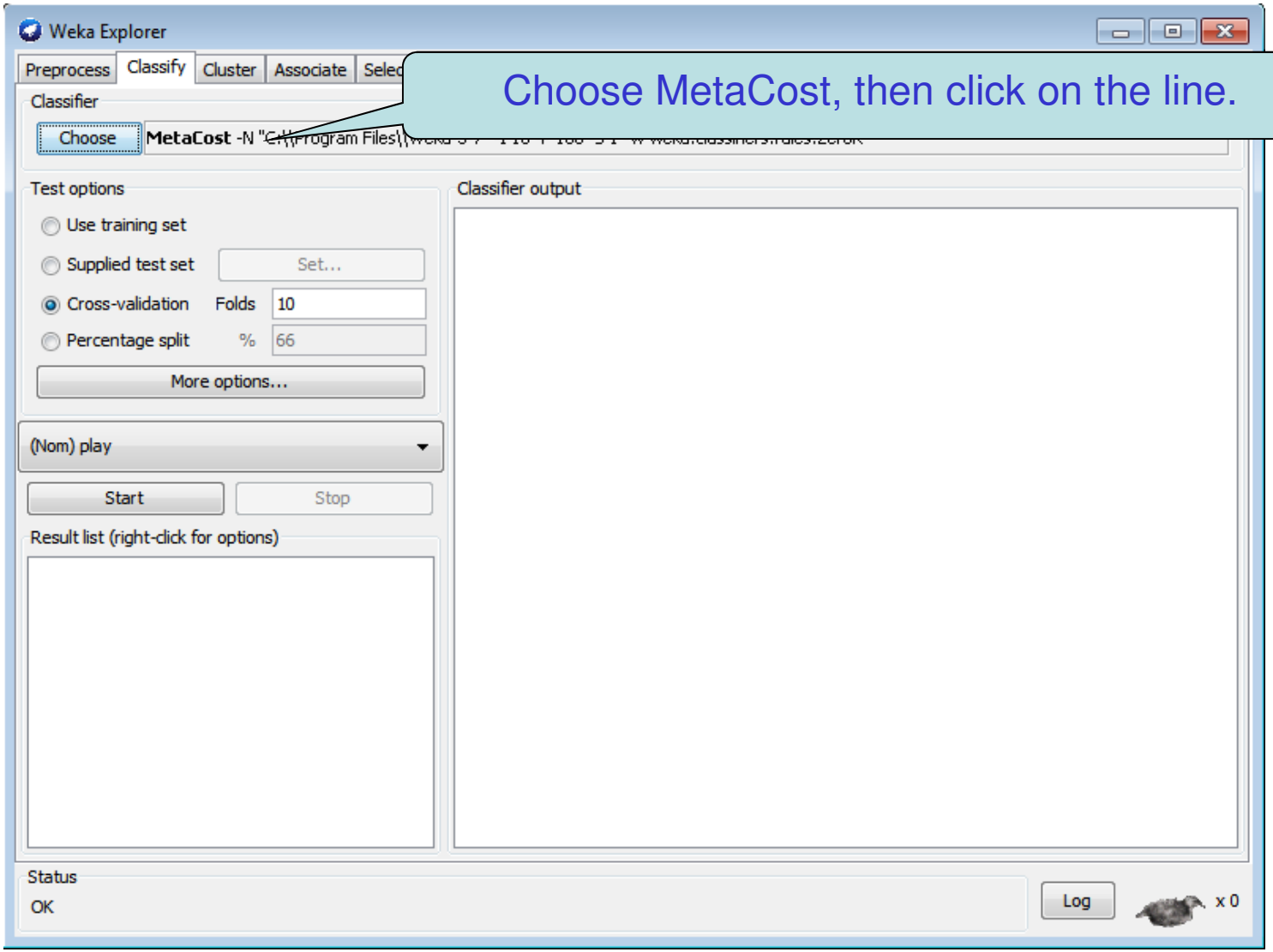
- Main idea is to **relabel** training instances based on

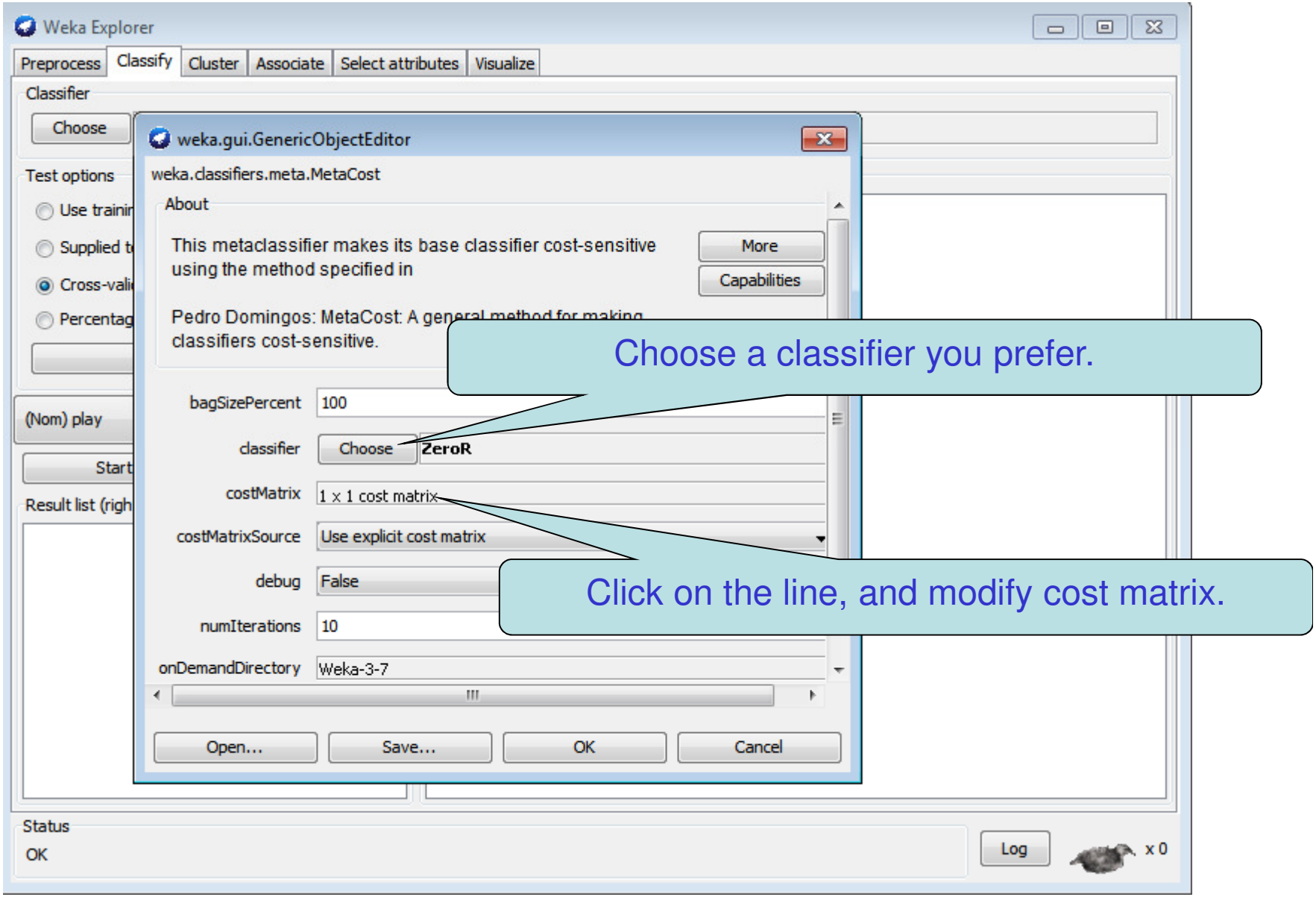
  $p(\text{n}|E)*c(\text{Y}, \text{n}) / c(\text{N}, \text{y}) < p(\text{y}|E)$

  then run the classifier on the modified training data.

- In short, MetaCost works as follows:
  - form multiple bootstrap replicates of the training set, and learn a classifier on each;
  - estimate each class's probability for each instance by the fraction of votes that it receives from the classifiers;
  - use the above equation to re-label each training instance with the estimated optimal class;
  - Finally, re-learn the classifier using relabeled training set.

Weka


Weka Explorer
Preprocess
Classify
Cluster
Associate
Classifier
Choose
MetaCost -N
Choose MetaCost, then click on the line.
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
Classifier output
(Nom) play
Start
Stop
Result list (right-click for options)
Status
OK
Log
x 0


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
Test options
Use trainin
Supplied t
Cross-vali
Percentag
(Nom) play
Start
Result list (righ
Status
OK
Log
x 0
weka.gui.GenericObjectEditor
weka.classifiers.meta.MetaCost
About
This metaclassifier makes its base classifier cost-sensitive using the method specified in
Pedro Domingos: MetaCost: A general method for making classifiers cost-sensitive.
More
Capabilities
bagSizePercent
100
classifier
Choose
ZeroR
costMatrix
1 x 1 cost matrix
costMatrixSource
Use explicit cost matrix
debug
False
numIterations
10
onDemandDirectory
Weka-3-7
Open...
Save...
OK
Cancel
Choose a classifier you prefer.
Click on the line, and modify cost matrix.