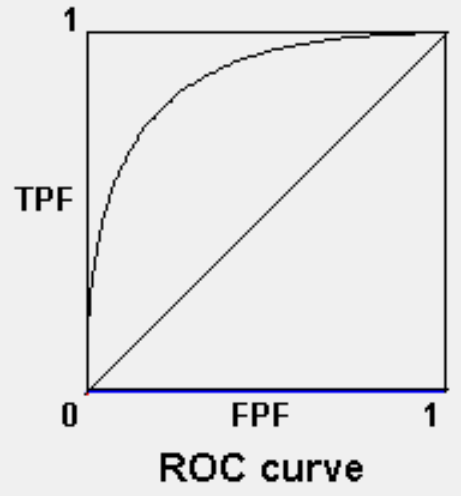
ROC Curves
Precision and Recall
Cumulative Gains Charts
Lift Charts
Cost-Based Classification

ROC Curves

ROC (Receiver Operating Characteristic) curve

- ROC curves were developed in the 1950's as a by-product of research into making sense of radio signals contaminated by noise. More recently it's become clear that they are remarkably useful in decision-making.
- They are a performance graphing method.
- **True positive** and **False positive** fractions are plotted as we move the dividing threshold of classification. They look like:



Confusion Matrix

	P	**N**
P	#True Positives (TP)	#False Negatives (FN)
N	#False Positives (FP)	#True Negatives (TN)

Predicted class

Actual class

Because of cross-validation, if there are *n* training instances, WEKA will generate a confusion matrix whose entries sum-up to *n*. This is because every training instance is used once as a test instance (when it is included in the test fold)

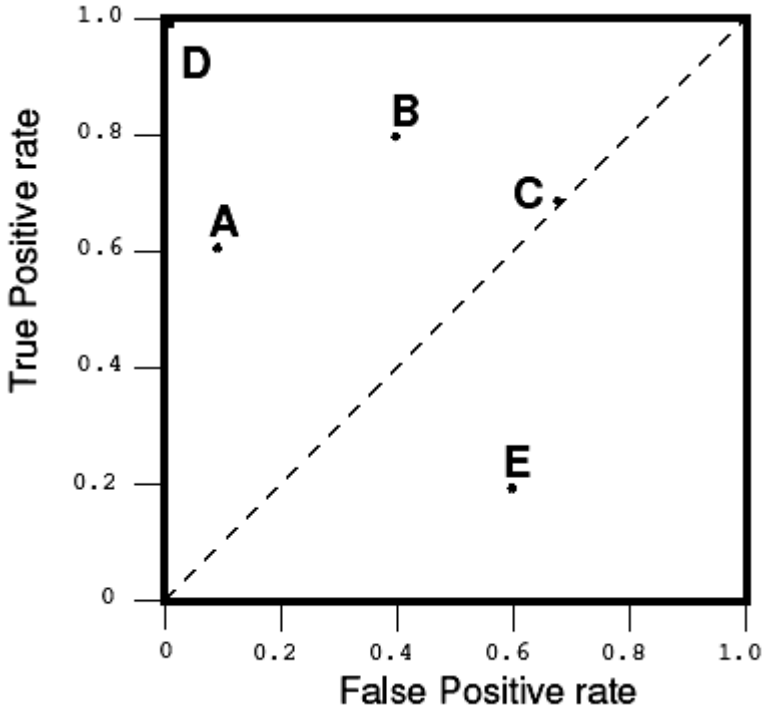
True positives and False positives

True positive rate is
TPR = P correctly classified / P = TP / P

False positive rate is
FPR = P incorrectly classified / N = FP / N

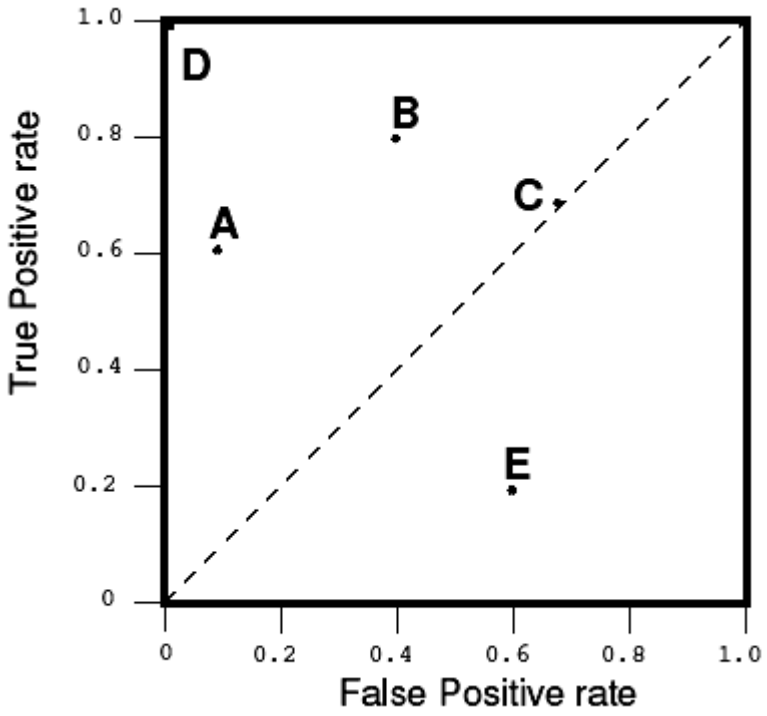
ROC Space

- ROC graph is two-dimensional graph in which TPR is plotted on the Y axis and FPR is plotted on the X axis.
- **ROC graph** depicts relative trade-offs between benefits (true positives) and costs (false positives).
- Figure shows an ROC graph with **five** classifiers labeled A through E.
- **Discrete classifier** is one that outputs only a class label.
- **Each discrete classier** produces **one** (fpr, tpr) pair corresponding to a **single point in ROC space**.
  - Classifiers in figure are all discrete classifiers.



Several Points in ROC Space

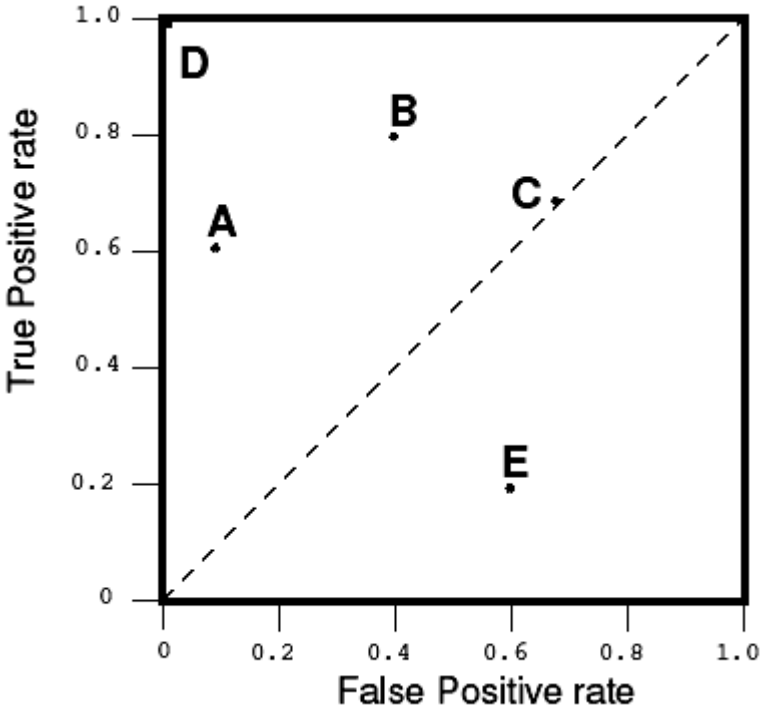
- Lower left point (0, 0) represents the strategy of never issuing a positive classification;
  - such a classier commits no false positive errors but also gains no true positives.
- Upper right corner (1, 1) represents the opposite strategy, of unconditionally issuing positive classifications.
- Point (0, 1) represents perfect classification.
  - D's performance is perfect as shown.
- Informally, one point in ROC space is better than another if it is to the northwest of the first
  - tp rate is higher, fp rate is lower, or both.



“Conservative” vs. “Liberal”

- Classifiers appearing more on the “west” may be thought of as “conservative”
  - they make positive classifications only with strong evidence so they make few false positive errors,
  - but they often have low true positive rates as well.

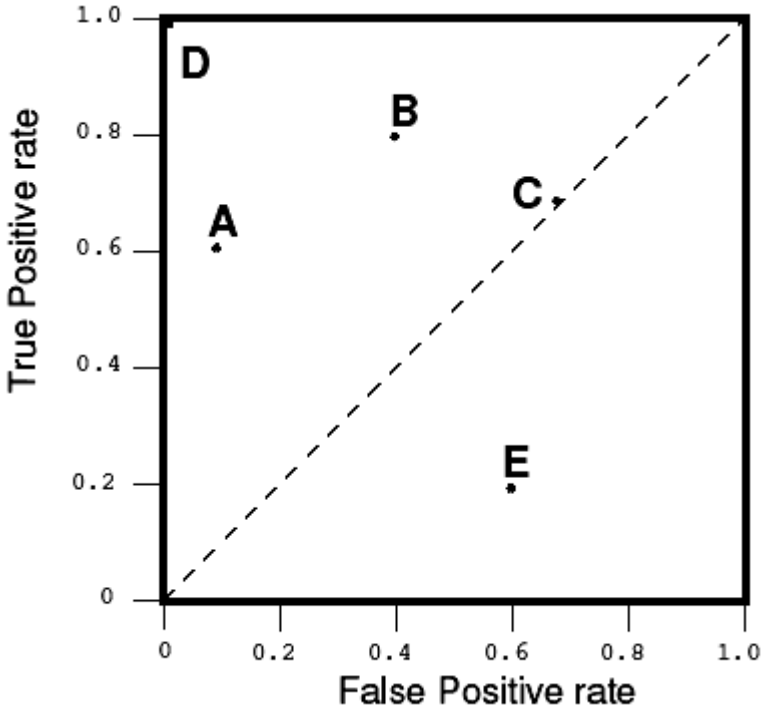
- Classifiers appearing more on the “north-east” may be thought of as “liberal”
  - they make positive classifications with weak evidence so they classify nearly all positives correctly,
  - but they often have high false positive rates.



- In figure, A is more conservative than B.

Random Performance

- The diagonal line y = x represents the strategy of randomly guessing a class.
- For example, if a classier randomly says “Positive” half the time (regardless of the instance provided), it can be expected to get half the positives and half the negatives correct;
  - this yields the point (0.5; 0.5) in ROC space.
- If it randomly say “Positive” 90% of the time (regardless of the instance provided), it can be expected to:
  - get 90% of the positives correct, but
  - its false positive rate will increase to 90% as well, yielding (0.9; 0.9) in ROC space.
- A random classier will produce a ROC point that "slides" back and forth on the diagonal based on the frequency with which it guesses the positive class.



C's performance is virtually random.

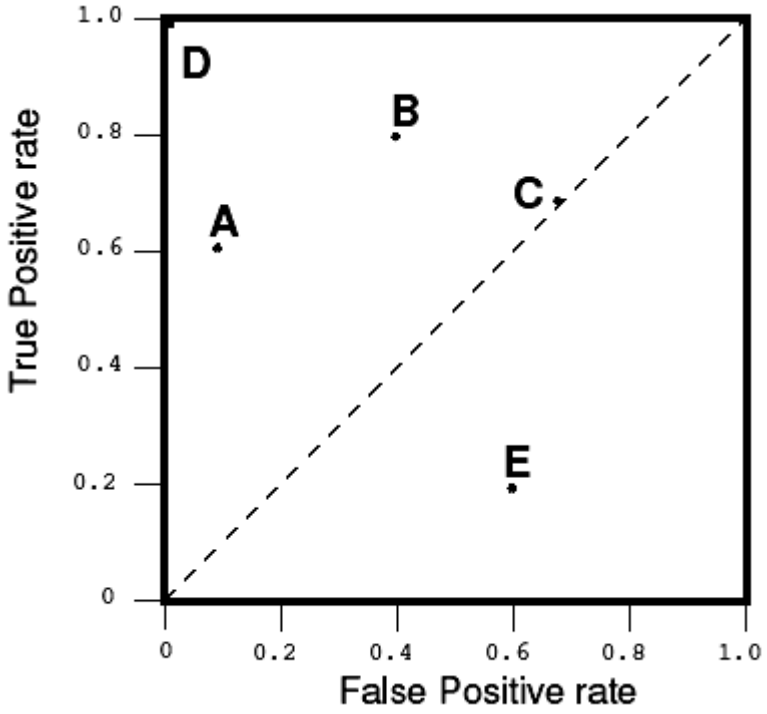
At (0.7; 0.7), C is guessing the positive class 70% of the time.

Upper and Lower Triangular Areas

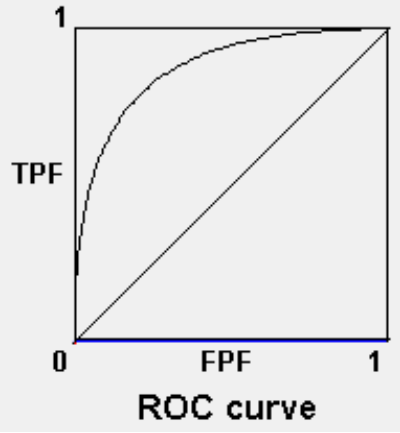
- To get away from the diagonal into the upper triangular region, the classifier must exploit some information in the data.

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

- If we negate a classifier that is, reverse its classification decisions on every instance, then:
  - its true positive classifications become false negative mistakes, and
  - its false positives become true negatives.
- A classifier below the diagonal may be said to have useful information, but it is applying the information incorrectly



Curves in ROC space

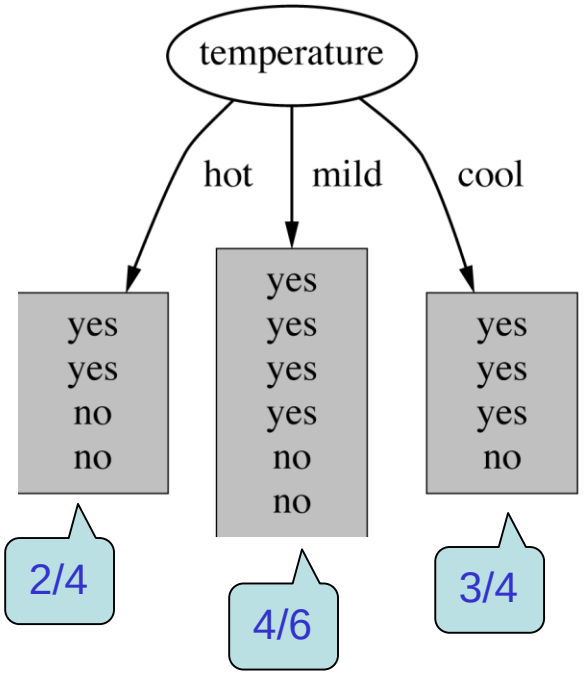


- Many classifiers, such as decision trees or rule sets, are designed to produce only a class decision, i.e., a Y or N on each instance.
  - When such a discrete classifier is applied to a test set, it yields a single **confusion matrix**, which in turn corresponds to **one** ROC point.

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

Creating Scoring Classifiers

- E.g, 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.
- For SVM build a scoring model on the distances of the test points to the separating hyperplane.
  - There is a log.reg. option in Weka for SMO
    - Distances will serve as a single predictor attribute for generating Log. Reg. probabilities for the class values.



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.

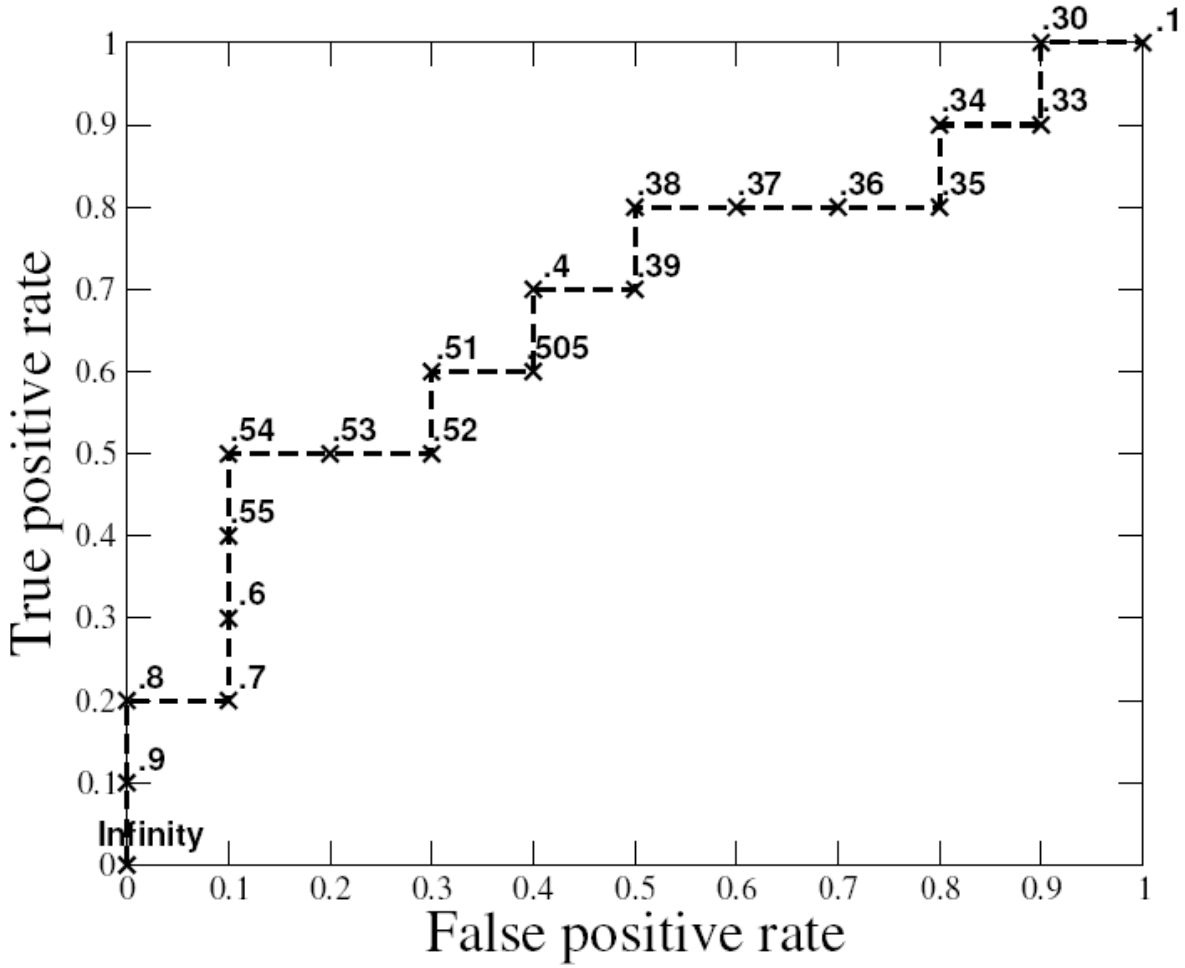
- Therefore, we can simply:
  - sort the test instances decreasing by their scores and
  - move down the list (lowering the threshold), processing one instance at a time and
  - update TPR and FPR as we go.

- In this way, an ROC graph can be created with just a linear scan.

Example of a test set and the scores

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



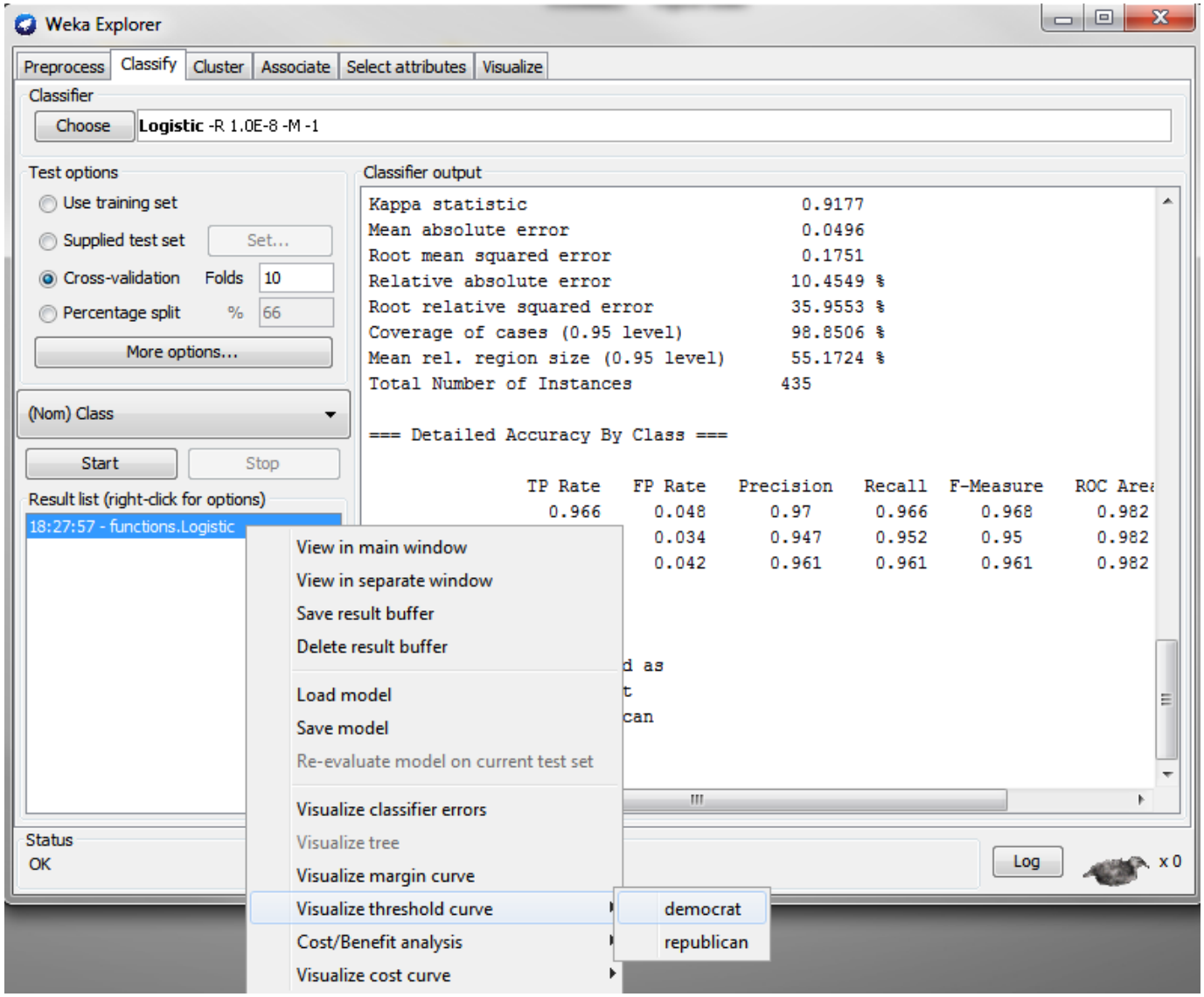
Question: Why do we have increments of 0.1 for both TPR and FPR? When would the increments be different?

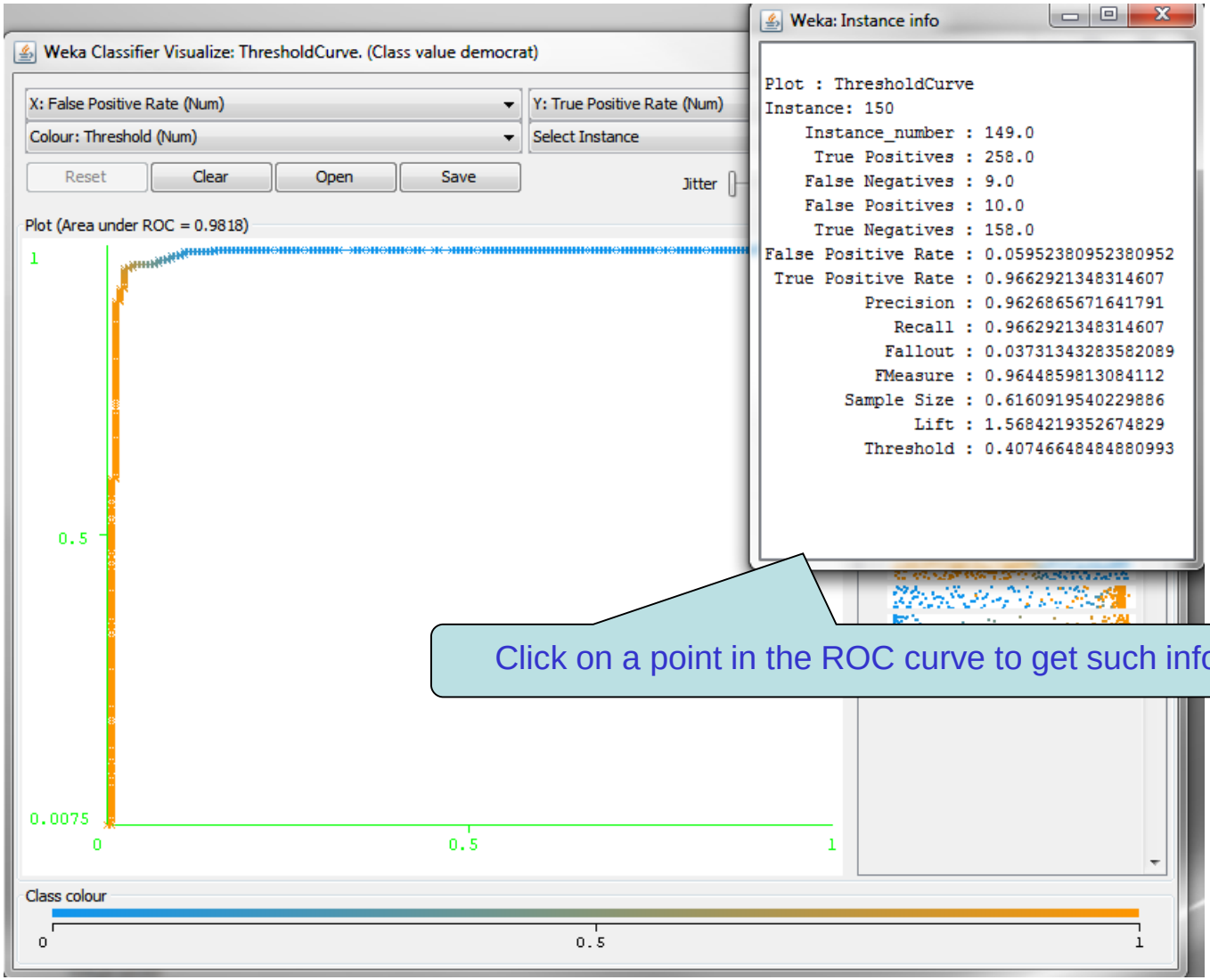
A threshold of +inf produces the point (0; 0).

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.

Lowering this threshold corresponds to moving from the "conservative" to the "liberal" areas of the graph.


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
Logistic -R 1.0E-8 -M -1
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
Classifier output
Kappa statistic 0.9177
Mean absolute error 0.0496
Root mean squared error 0.1751
Relative absolute error 10.4549 %
Root relative squared error 35.9553 %
Coverage of cases (0.95 level) 98.8506 %
Mean rel. region size (0.95 level) 55.1724 %
Total Number of Instances 435
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure ROC Area
0.966 0.048 0.97 0.966 0.968 0.982
0.034 0.947 0.952 0.95 0.982
0.042 0.961 0.961 0.961 0.982
(Nom) Class
Start
Stop
Result list (right-click for options)
18:27:57 - functions.Logistic
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
democrat
republican
Status
OK
Log
x 0


Weka Classifier Visualize: ThresholdCurve. (Class value democrat)
X: False Positive Rate (Num)
Y: True Positive Rate (Num)
Colour: Threshold (Num)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot (Area under ROC = 0.9818)
1
0.5
0.0075
0
0.5
1
Class colour
0
0.5
1
Weka: Instance info
Plot : ThresholdCurve
Instance: 150
Instance_number : 149.0
True Positives : 258.0
False Negatives : 9.0
False Positives : 10.0
True Negatives : 158.0
False Positive Rate : 0.05952380952380952
True Positive Rate : 0.9662921348314607
Precision : 0.9626865671641791
Recall : 0.9662921348314607
Fallout : 0.03731343283582089
FMeasure : 0.9644859813084112
Sample Size : 0.6160919540229886
Lift : 1.5684219352674829
Threshold : 0.40746648484880993
Click on a point in the ROC curve to get such information.

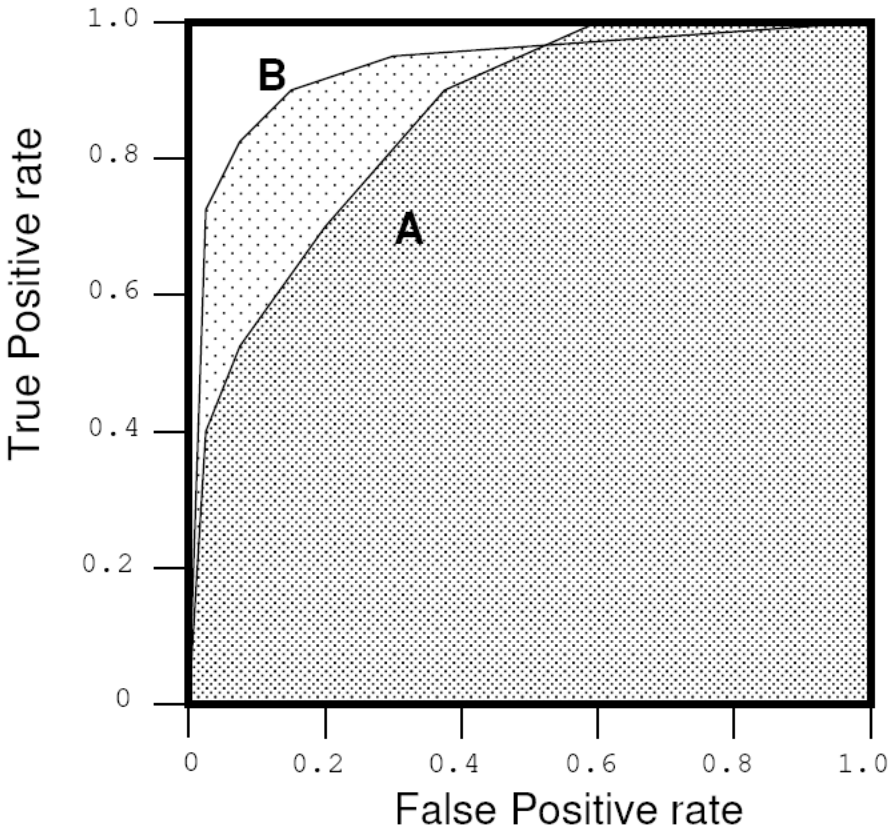
Interpretation

- Each point in the WEKA ROC curve corresponds to classifier (which we get by setting a threshold)
- In other words, each point corresponds to a confusion matrix.
  - Recall, because of cross-validation, if there are *n* training instances, WEKA will generate a confusion matrix whose entries sum-up to *n*. This is because every training instance is used once as a test instance (when it is included in the test fold)

Area under an ROC Curve

- AUC has an important statistical property:
  - The AUC of a classifier is equivalent to the probability that the classifier will rank a **randomly chosen positive** instance higher than a **randomly chosen negative** instance.
  - The bigger AUC the better.

- AUC can be computed by a slight modification to the algorithm for constructing ROC curves.



Precision and Recall

- **Precision** = TP / (TP+FP)
- **Recall** = TP / P (same as TPR)

Weka also produces fallout.
Fallout = FP / (TP + FP)

- **F measure** is the harmonic mean of the two:
  - Fmeasure = 2 / [(1/Precision) + (1/Recall)] =
    2*Precision*Recall / (Precision + Recall)

- Example: A dumb classifier that always say "Yes" has a 100% recall, but very bad precision (e.g. 10% is there are only 10% positives in the test dataset).

- Example: Consider now a classifier that has a recall of 90% and a precision of 40%.

- The F measure for the first classifier is: 2*1*0.1/(1+0.1) ~0.18
- The F measure for the second classifier is: 2*0.9*0.4/(0.9+0.4) ~0.55
- So, the F measure clearly shows that the second classifier is much better.

Harmonic Mean

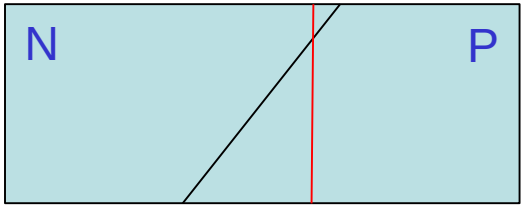
- Harmonic mean is closer to the lower numbers.

- **Use** the **harmonic mean** when your sample contains fractions and/or extreme values (either too big or too small). It is more stable regarding outliers.

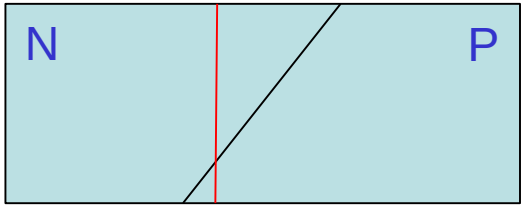
- For example: the arithmetic **mean** of (1,2,3,4,5,100) is 19.2 whereas the **harmonic** one is 2.58

Precision and Recall Examples

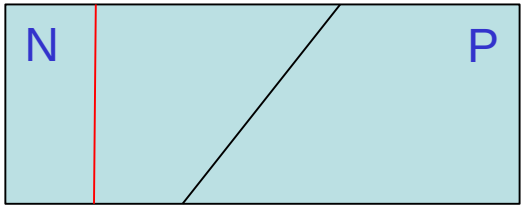
The red line is the classification line. Everything on the right is classified as "P". Everything on the left is classified as N.



Low recall, high precision.



Higher recall, lower precision.



Highest recall (100%), quite low precision.

Sensitivity and Specificity

Popular in the medical domain.

- **Sensitivity** = TP / P (same as TPR and Recall)
  - Example: Fraction of patients with cancer that the classifier rightly predicts.
- **Specificity** = TN / N
  - Example: Fraction of patients without cancer that the classier rightly rules out.

- Often doctors like to increase sensitivity at the expense of specificity.
  - Typically, achieved by lowering the threshold for predicting "Yes", i.e. going more on the right of the ROC curve.

Cumulative Gains Chart, Lift Chart

Decisions

Example: promotional mailout to 1,000,000 households

Mail to all; 0.1% respond (1000)

- Data algorithm identifies a subset of 100,000 most promising, 0.4% of these (i.e. 400) are expected to respond

  40% of the 1000 responses (we would get) for just 10% of cost may pay off

- Also, the same algorithm identifies a subset of 400,000 most promising, 0.2% of these (i.e. 800) are expected to respond

  80% of the 1000 responses (we would get) for 40% of cost may still pay off

Model-Sorted List

Use a DM algorithm (M) to assign a score to each (testing data) customer
Sort customers by decreasing score
Expect more targets (hits) near the top of the list

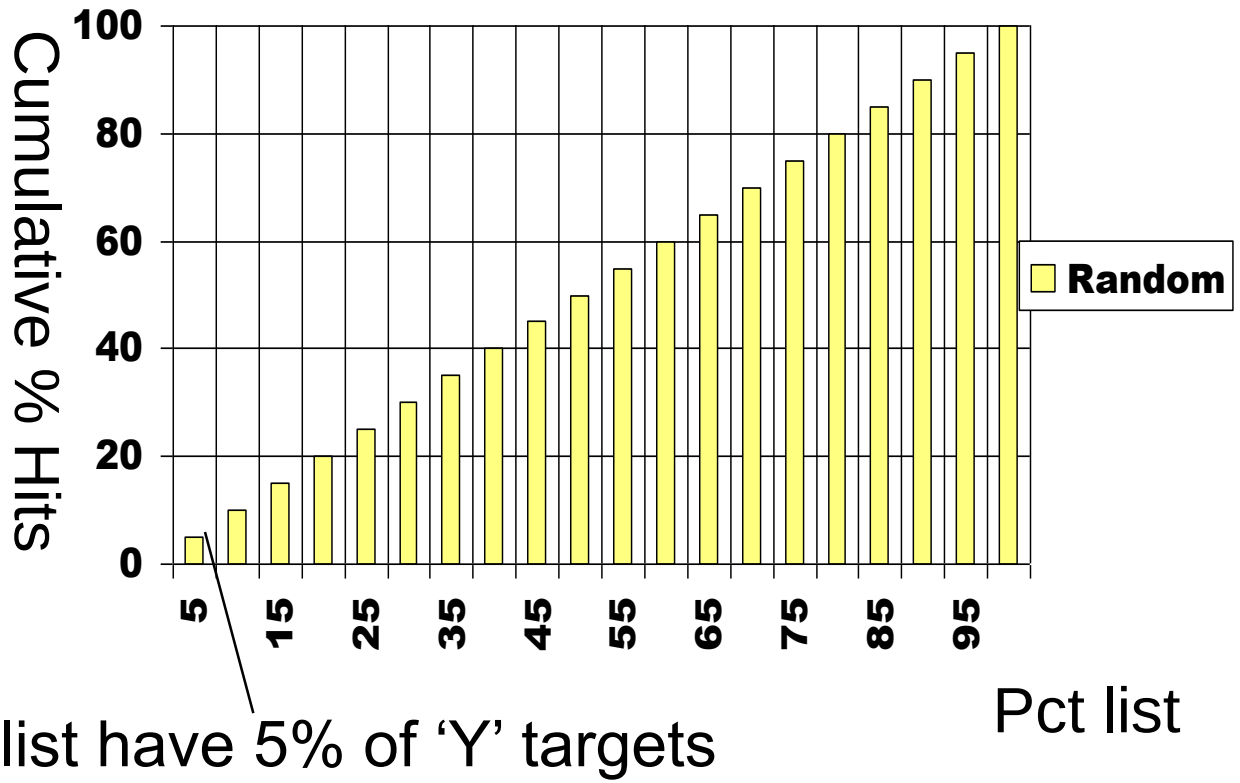
No	Score	Target	CustID	Age	
1	0.97	Y	1746	…	
2	0.95	N	1024	…	
3	0.94	Y	2478	…	
4	0.93	Y	3820	…	
5	0.92	N	4897	…	
…	…		…	…	
80	0.06	N	2422		

3 hits in top 5% of the list

If there are, say, 15 “Yes” (testing data customers) overall, then top 5% has 3/15=20% of ‘Yes’ targets

CPH (Cumulative Pct Hits)

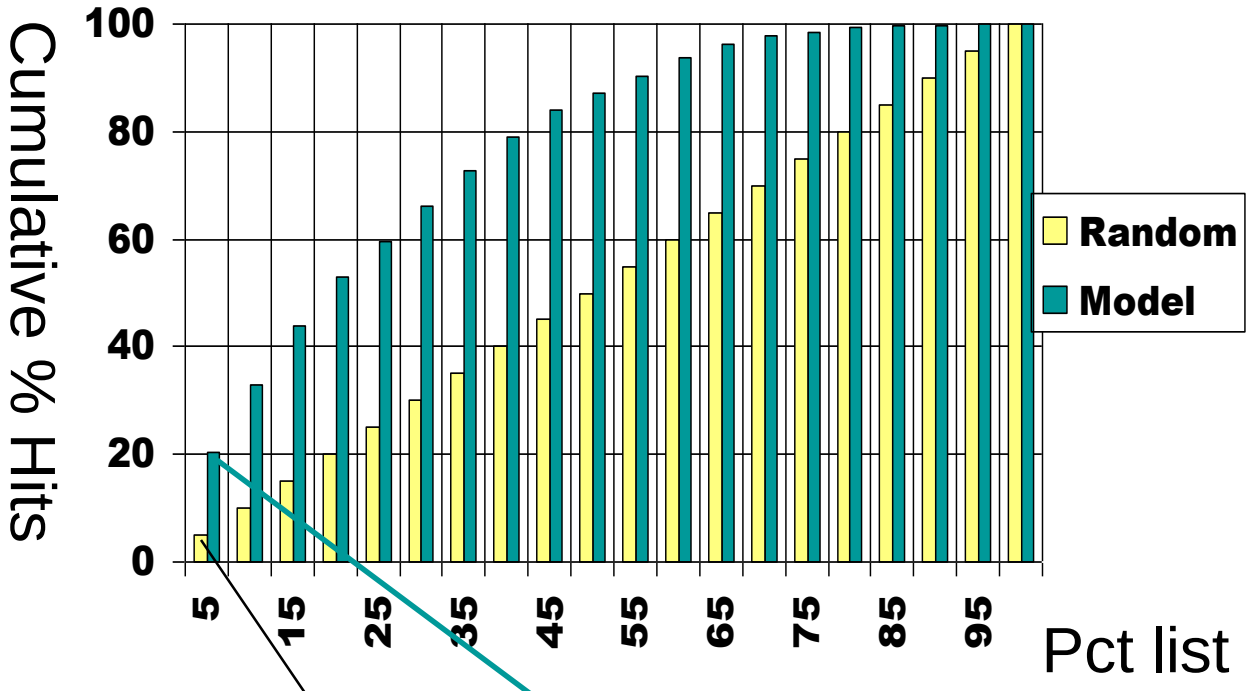
CPH(P,M)=% of 'Y' targets in the first **P%** of the list scored by model **M**



Q: What is expected value for CPH(P,Random) ?

A: Expected value for *CPH(P,Random) = P*

CPH: Random List vs Model-ranked list



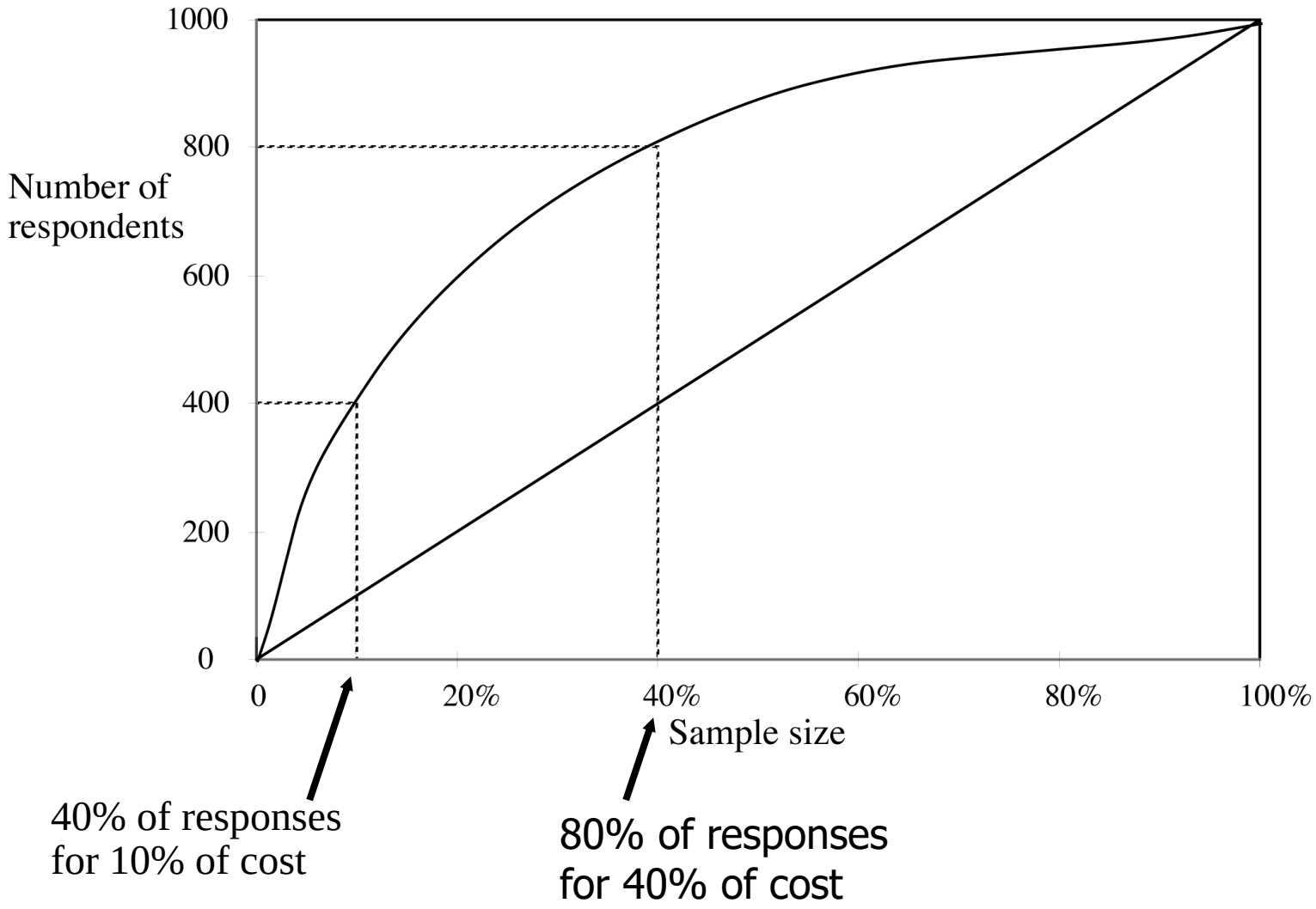
5% of random list have 5% of 'Y' targets,

but 5% of DM Algo. ranked list have 21% of 'Y' targets

CPH(5%,model)=21%.

Cumulative gains chart

This time, instead of pct's we show real number of positive "Y" people.



Lift chart

Lift(P,M) = CPH(P,M) / P

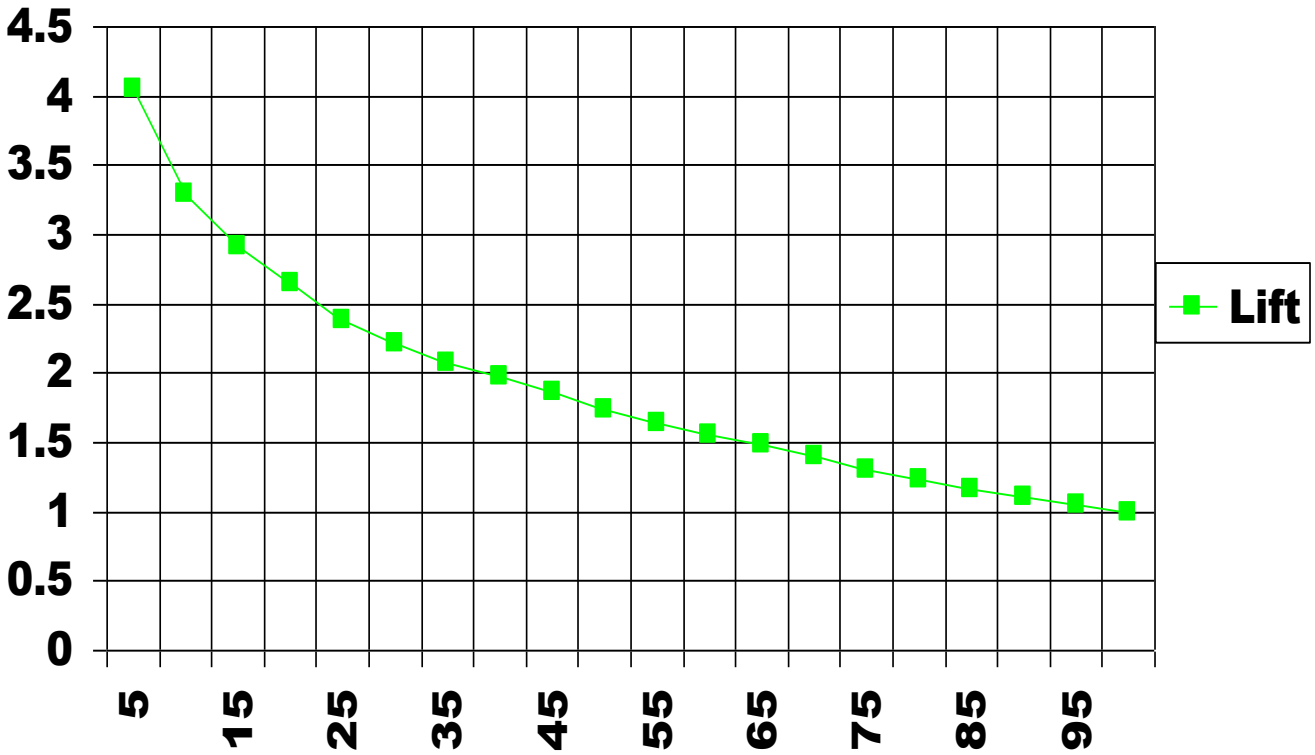
Lift (at 5%)

= 21% / 5%

= 4.2

better

than random



P -- percent of the list

Cost-based Classification

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 nontrivial solution is sought.

- Evaluation by classification accuracy also tacitly assumes equal error costs---that 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.

Cost based classification

- Let {y,n} be the positive and negative class values.
- 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

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

E.g. suppose we are classifying for cancer.
c(Y,n) = 1 and c(N,y)=10
Now, suppose
p(n|E) = 0.8 and p(y|E) = 0.2

Without costs, we predict "N".
With costs, we predict "Y".
0.8 * (1/10) < 0.2

Cost-based Classification

MetaCost (Domingos, 1999)

- Main idea is to relabel training instances using

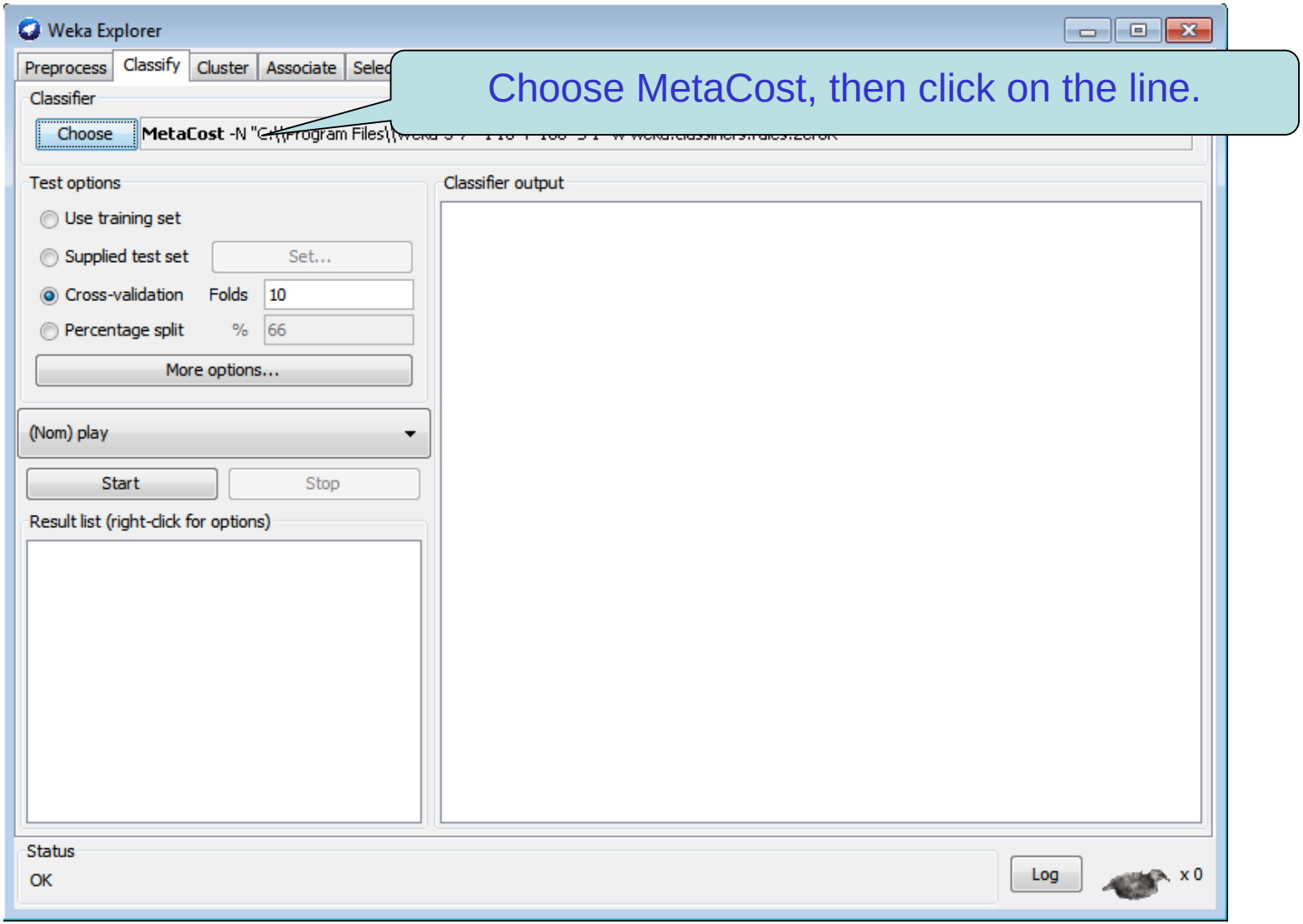
  $p(\text{n}|E) * c(\text{Y},\text{n}) / c(\text{N},\text{y}) < p(\text{y}|E)$

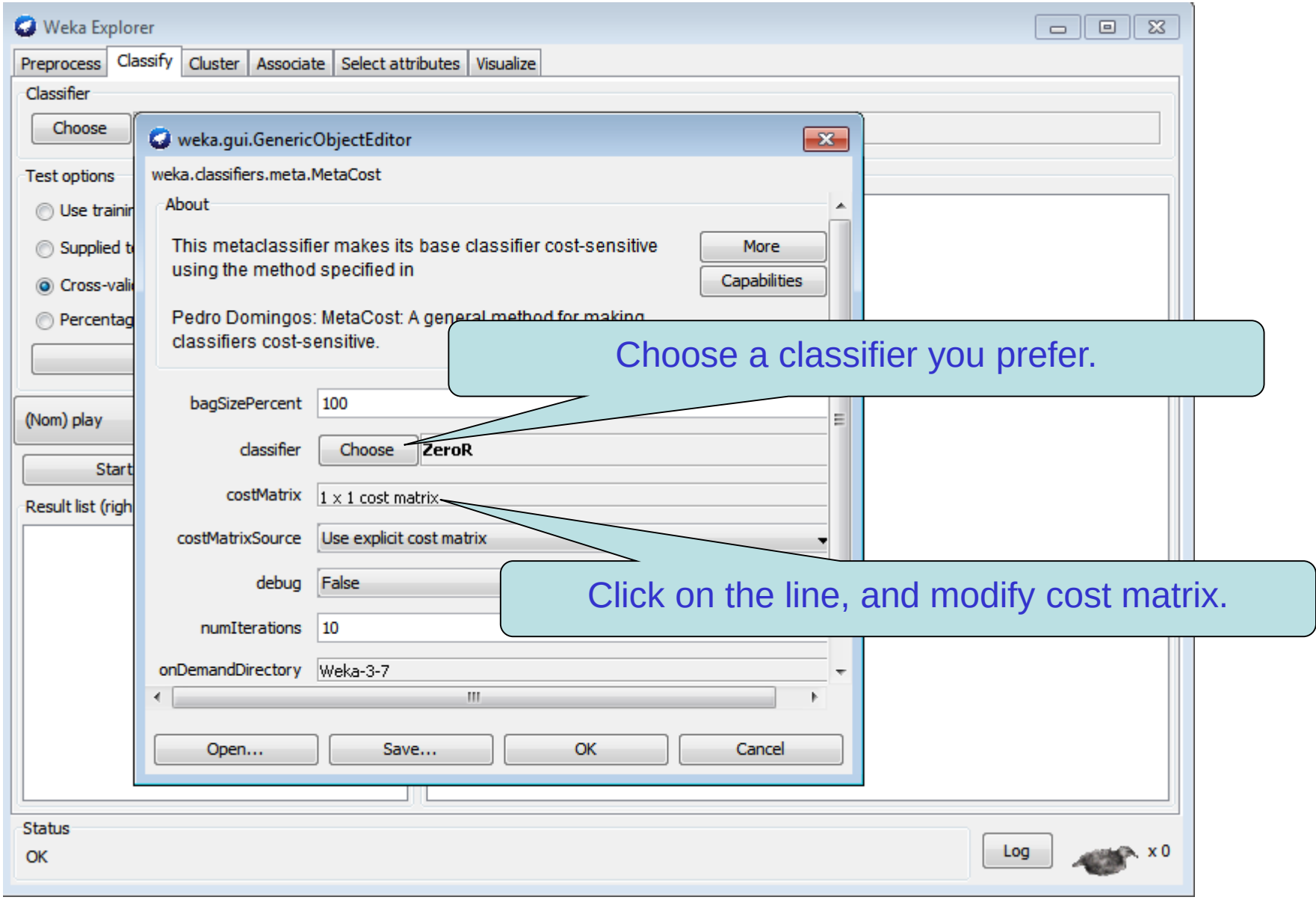
  i.e. assume we don't know the class of training examples and classify them using some simple algorithm, e.g. Naïve Bayes, using the above inequality.

- In this way some of the instances can change class. Most of those will be (relatively weak) negative instances that were classified positive.

- Then run any algorithm you prefer on the modified training data.

Weka


Weka Explorer
Preprocess
Classify
Cluster
Associate
Choose MetaCost, then click on the line.
Classifier
Choose
MetaCost -N
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.