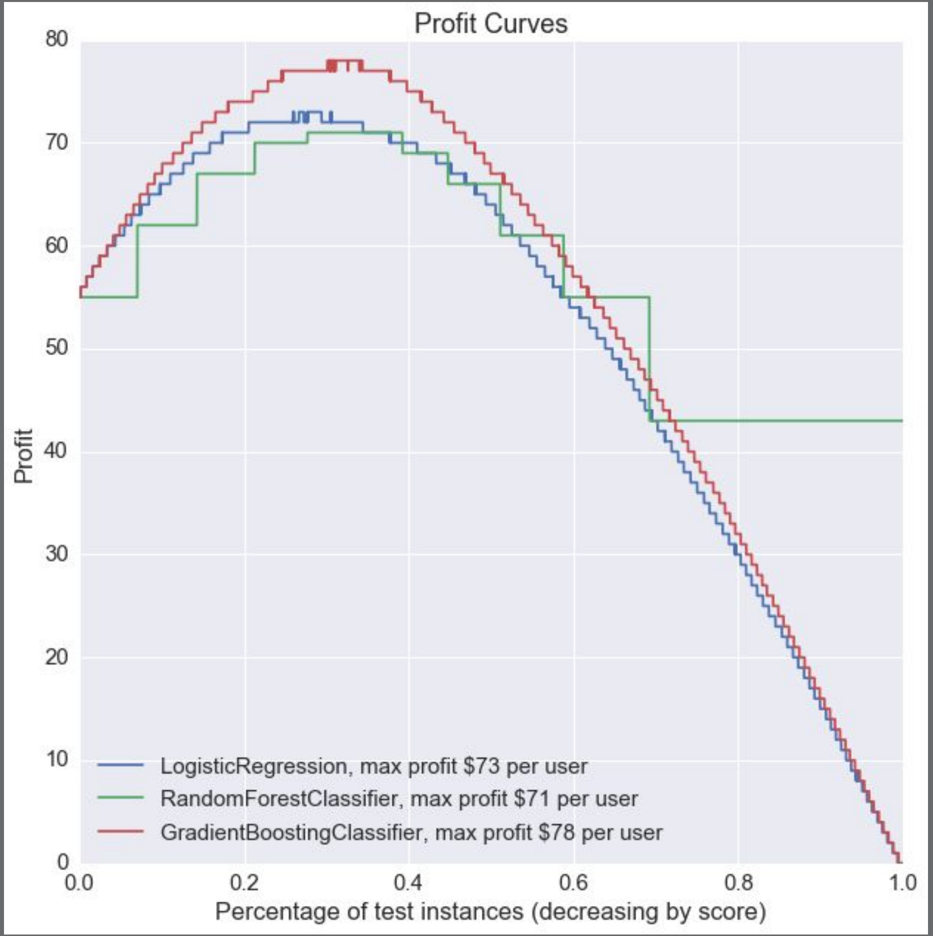
Profit Curves and Imbalanced Classes

Taryn Heilman
Frank Burkholder
Last modified: November 7, 2019

galvanize



Learning Objectives

- **Review:** confusion matrices and classification metrics
- **Discuss** challenges with imbalanced datasets
- **Learn, compare, and be ready to implement** 3 methods of dealing with imbalanced data:
  - Practical steps (tweak your model)
  - Cost sensitive learning (consider costs of FP, FN)
  - Sampling methods (balance your classes)

Review Relevant Concepts

- What is a confusion matrix?
- Describe the metrics you can calculate from a confusion matrix
- Give an example of when each metric might be most useful
- What is the default model score from a classifier? Why is this problematic?
- What is an ROC curve?

Review: Confusion Matrix

	Predicted False ($\hat{Y} = 0$)	**Predicted** True ($\hat{Y} = 1$)
Negative class ($Y = 0$)	True Negatives (TN)	False Positives (FP)
Positive class ($Y = 1$)	False Negatives (FN)	True Positives (TP)

There are many ways to evaluate the confusion matrix.
Here is one - the default model score for classifiers.

Accuracy: overall proportion correct

$$accuracy = \frac{TN + TP}{FP + FN + TN + TP}$$

```
# y_test, y_pred are 1D np arrays

acc = (y_test == y_pred).mean()
```

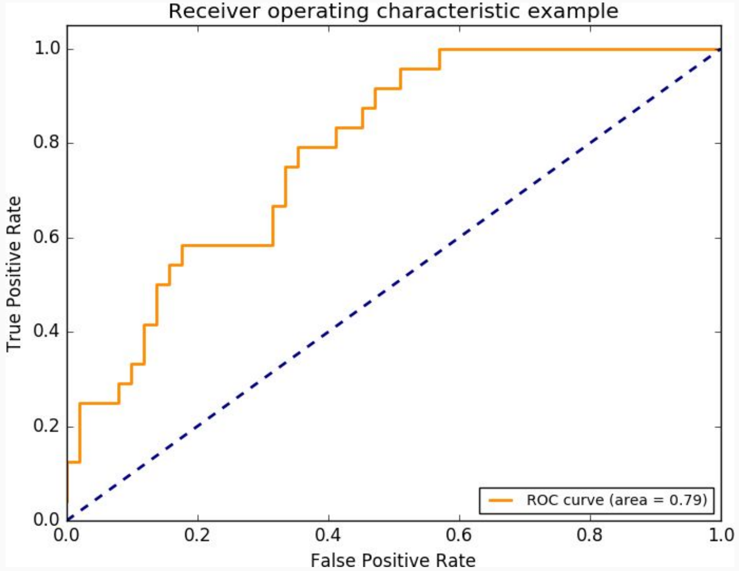
Review: Binary Classification Metrics

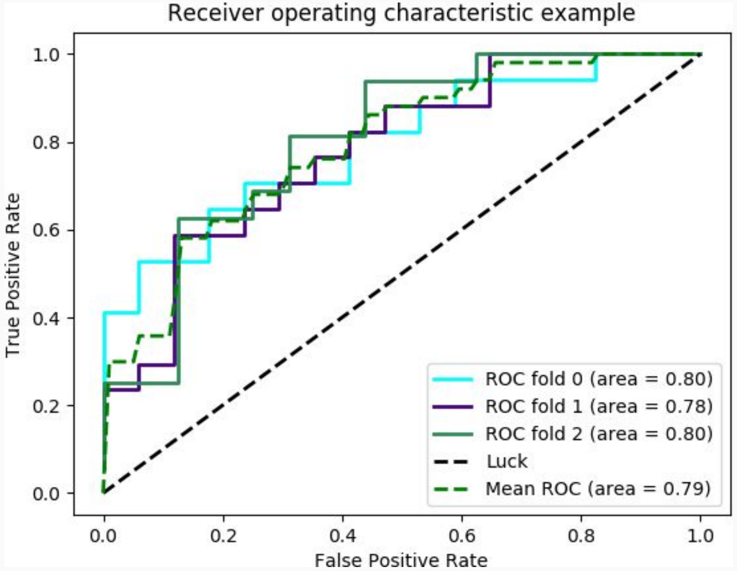
galvanize

Metric	Description	Formula
Accuracy	fraction of data correctly classified	$\frac{TN+TP}{FP+FN+TN+TP}$
Precision (Positive Predictive Value)	proportion called true that are correct	$\frac{TP}{FP+TP}$
True Positive Rate (Recall, Sensitivity)	proportion of positives that are labeled positive	$\frac{TP}{FN+TP}$
F1 score	balancing Precision/Recall	$\frac{2}{\frac{1}{recall}+\frac{1}{precision}}$
True Negative Rate (specificity)	fraction of negatives that are labeled negative	$\frac{TN}{TN+FP}$
False Positive Rate	1 - specificity	$\frac{FP}{TN+FP}$
False Negative Rate	1 - sensitivity	$\frac{FN}{TP+FN}$

https://en.wikipedia.org/wiki/Confusion_matrix

Review: ROC Curve





Plot TPR vs. FPR for every probability threshold

Metric is area under the curve (AUC), which tells us the probability that a randomly selected positive example will have a higher predicted probability than a randomly selected negative example

Motivation: Overheard in the Data Science Lounge...

“I just made the most amazing predictive model.”

“Do tell.”

“99.99% accuracy on fraud/not fraud test data.”

“What ratio of instances in the training data are fraud?”

“Fraud is super rare! Only 1 in 10,000!”

What would you ask next?

Problem Motivation

- Classification datasets can be “imbalanced”.
  - i.e. many observations of one class, few of another
  - Will give concrete examples later, but even a minority class of comprising 33% of the data can be considered imbalanced.

- Costs (in time, money, or life!) of a false positive is often different from cost of a false negative. Need to consider external (e.g. business) costs.
  - e.g. missing fraud can be more costly than screening legitimate activity
  - False negative in disease screening vs False negative in email spam classification

- Accuracy-driven models will over-predict the majority class.

Solutions

Practical steps (help your model fit better):

- Stratifying train_test_split
- Change weighting of training data for poorly represented class

Cost-sensitive learning (use outside costs & benefits to set prob. thresh):

- thresholding (aka “profit curves”)

Sampling (reduce imbalance with more/less data):

- Oversampling
- Undersampling
- SMOTE - **S**ynthetic **M**inority **O**versampling **TE**chnique

Dealing with imbalanced classes: Practical steps

galvanize

Practical steps - Stratifying train_test_split

If you have a minority class, are you sure it's represented in the same proportion in your y_train and y_test datasets?

```
X_train, X_test, y_train, y_test = train_test_split(X, y)
```

Maybe we'll get lucky!?

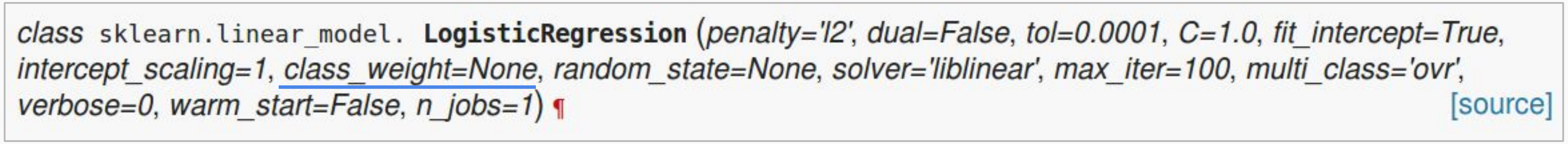
But this is better:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify = y)
```

Discuss: Can anyone think of a drawback here?

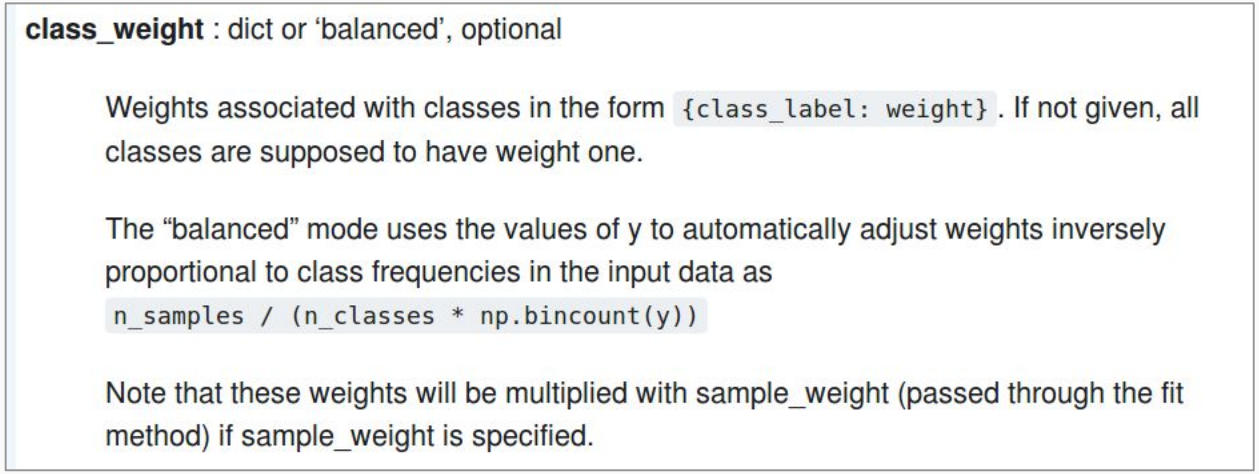
Practical steps - weight points of minority class

In objective function minimization, all classes are weighted equally by default:



class `sklearn.linear_model.` **`LogisticRegression`** (*penalty='l2', dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='liblinear', max_iter=100, multi_class='ovr', verbose=0, warm_start=False, n_jobs=1*) ¶ [source]

Option 1)



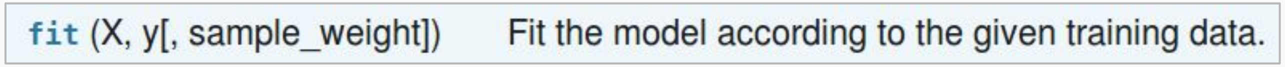
class_weight : dict or 'balanced', optional

Weights associated with classes in the form `{class_label: weight}`. If not given, all classes are supposed to have weight one.

The "balanced" mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data as `n_samples / (n_classes * np.bincount(y))`

Note that these weights will be multiplied with sample_weight (passed through the fit method) if sample_weight is specified.

Option 2)



`fit` (X, y[, sample_weight]) Fit the model according to the given training data.

Dealing with imbalanced classes: Cost sensitive learning - AKA profit curves

galvanize

Cost sensitive learning: basic idea

- Quantify relative costs of TP, FP, TN, FN

- Construct a confusion matrix for each probability threshold, and use a cost-benefit matrix to calculate a "profit" for each threshold. Pick the threshold that give the highest profit.

Cost-sensitive Learning - Thresholding

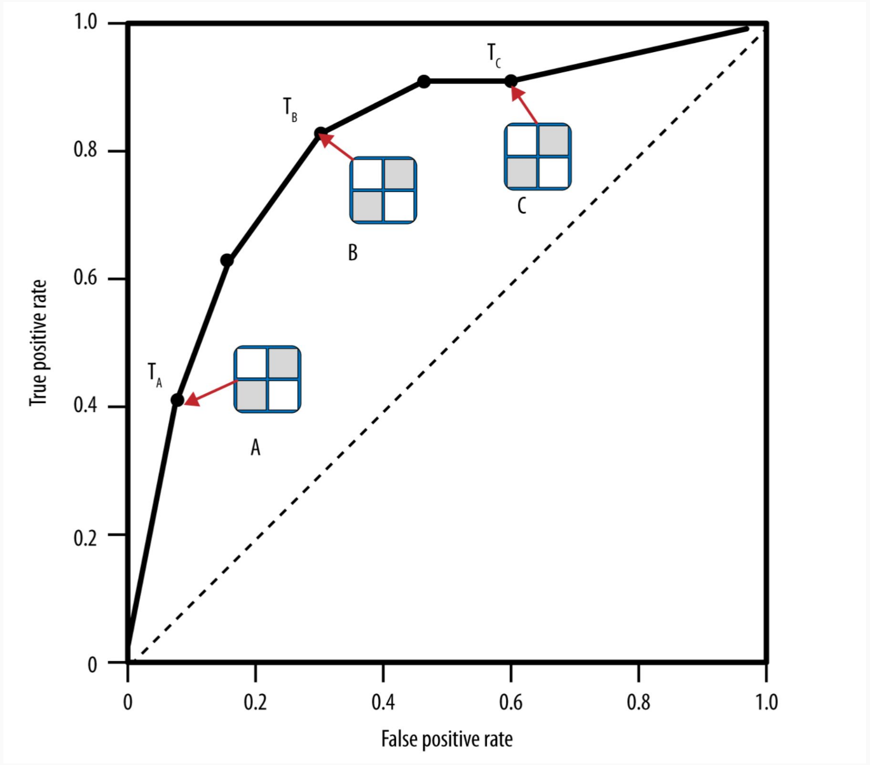
Recall the ROC Curve:

- ROC shows FPR = (1-TNR) vs TPR (aka Recall)
- doesn't give preference to one over the other

Q: How to handle unequal error costs?

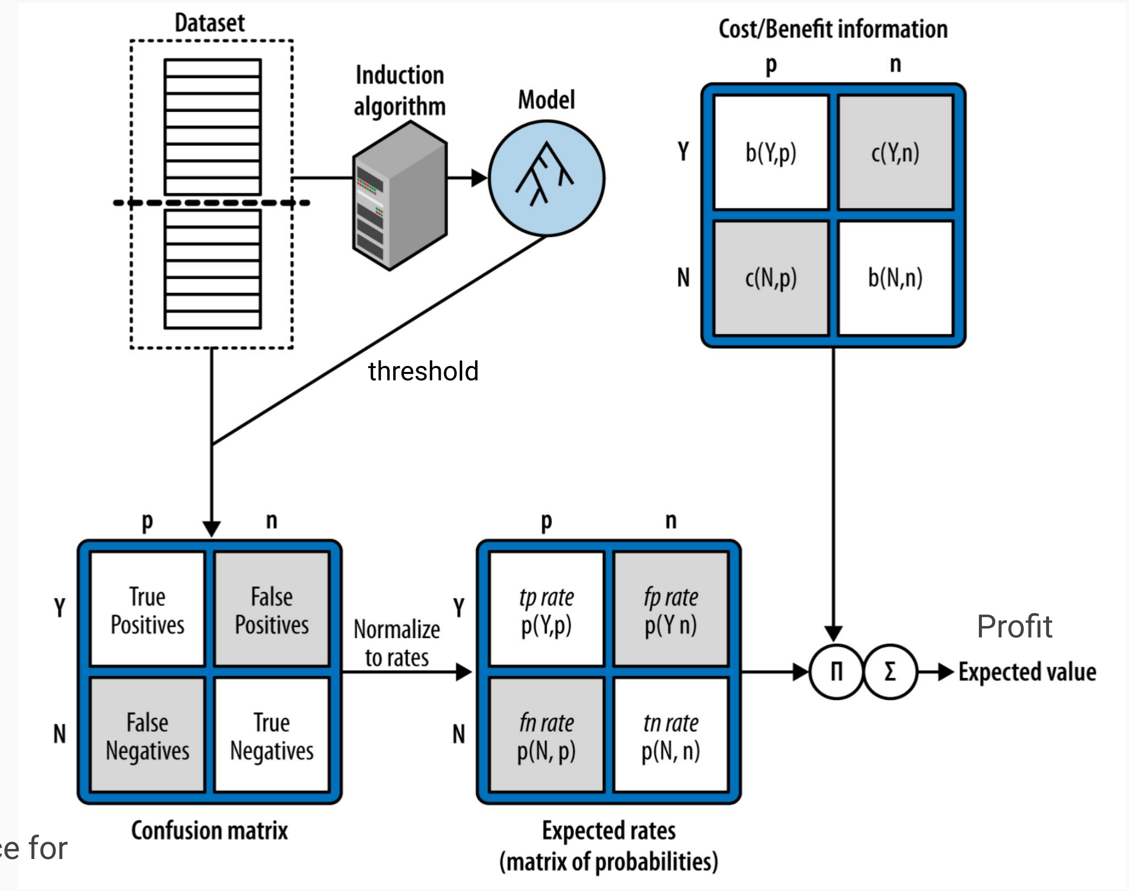
Q: Which threshold to pick? (assessment)

A: Plot expected profit (4 steps)!



Provost & Fawcett, Data Science for Business, O'Reilly 2013

Cost-sensitive Learning - Overview

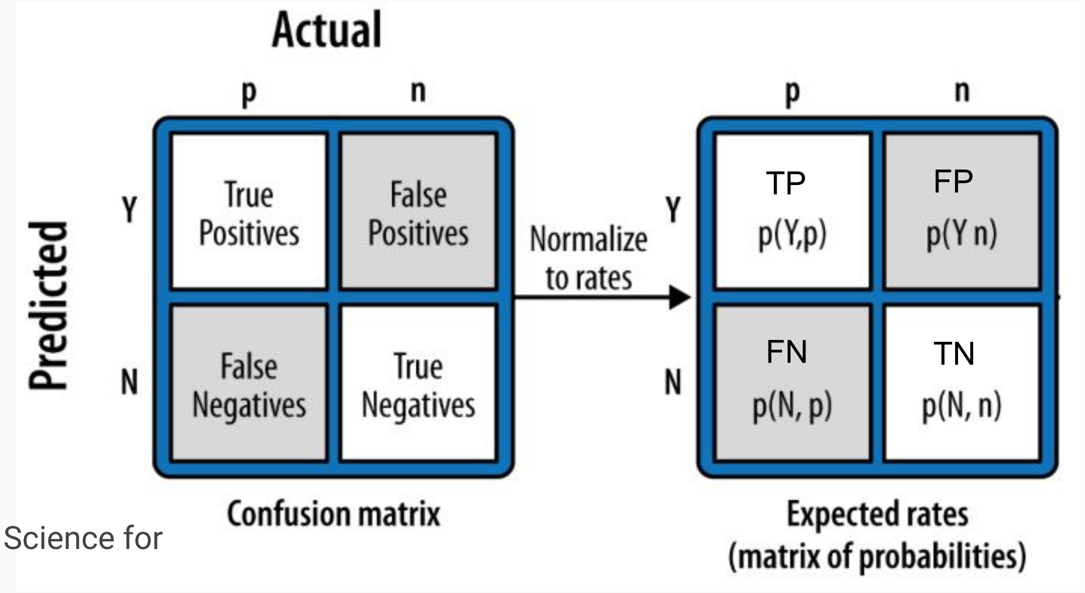


Provost & Fawcett, Data Science for Business, O'Reilly 2013

Cost-sensitive Learning - Thresholding

Computing Expected Profit

Step 1 - Estimate error probabilities based on a given threshold (will review thresholding in a bit)



Provost & Fawcett, Data Science for Business, O'Reilly 2013

Cost-sensitive Learning - Thresholding

Computing Expected Profit

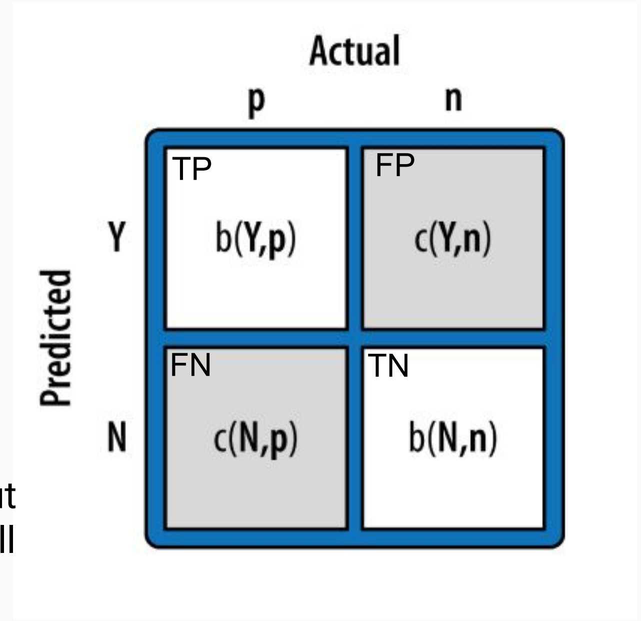
Step 2 - Define the cost-benefit matrix (based on your out-of-model knowledge)

Example
(mailing out survey to get consumer information)

$$\begin{array}{ccc} & & \text{Actual} \\ & & \begin{array}{cc} \mathbf{p} & \mathbf{n} \end{array} \\ \text{Predicted} & \begin{array}{c} \mathbf{Y} \\ \mathbf{N} \end{array} & \begin{pmatrix} 49 & -1 \\ 0 & 0 \end{pmatrix} \end{array}$$

\$50 - \$1

Costs \$1 to mail survey, but if we get it back we can sell information in it for \$50!



Predicted \ Actual	p	n
Y	TP b(Y,p)	FP c(Y,n)
N	FN c(N,p)	TN b(N,n)

Cost-sensitive Learning - Thresholding

Computing Expected Profit

Step 3 - Combine probabilities and payoffs.

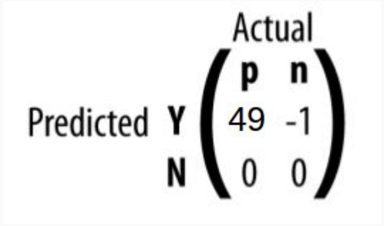
TP	FP
FN	TN

Cost-sensitive Learning - Thresholding and Profit Curve

galvanize

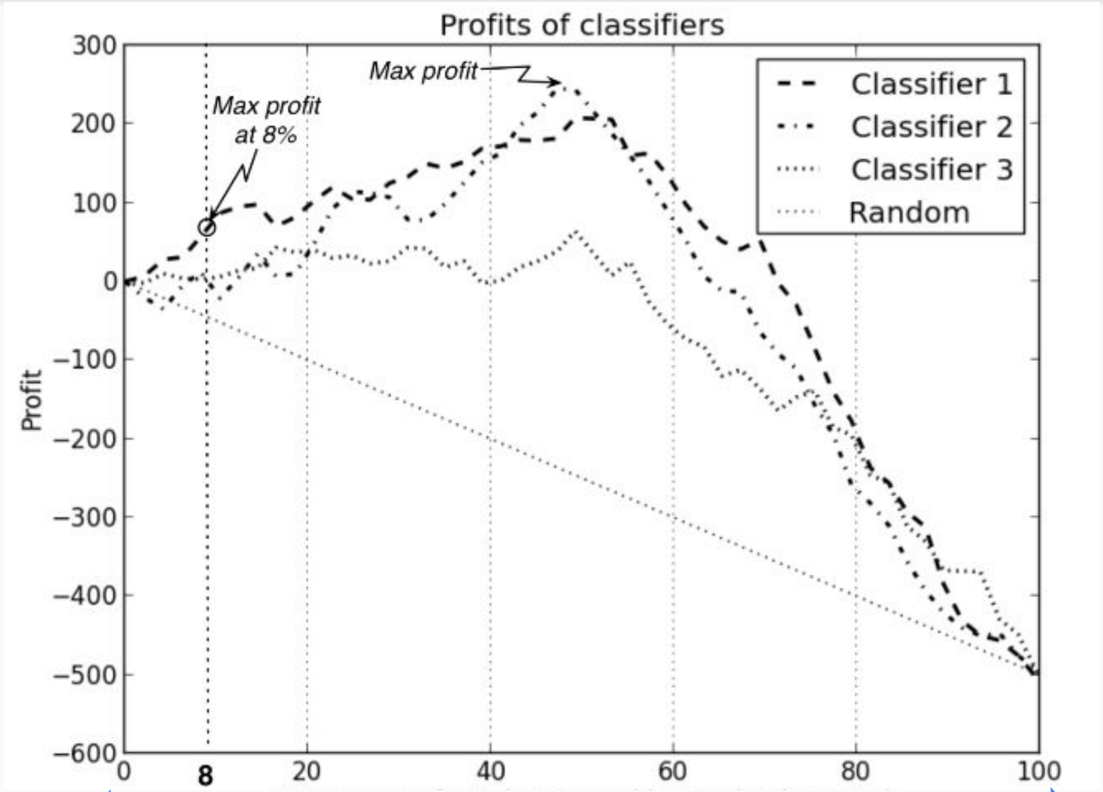
Step 4 - Plot profit

Cost - benefit matrix

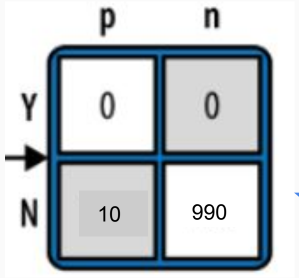


$$\begin{array}{ccc} & \multicolumn{2}{c}{\text{Actual}} \\ & p & n \\ \text{Predicted } Y & 49 & -1 \\ N & 0 & 0 \end{array}$$

Note no profit / cost on this matrix for FN, TN



Percent of test instances (decreasing by score)

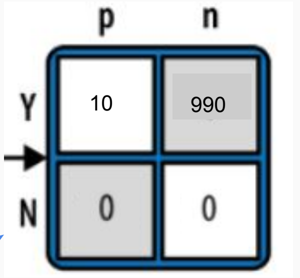


	p	n
Y	0	0
N	10	990

Threshold = 1.0

<- Sorted by threshold ->

Threshold = 0.0

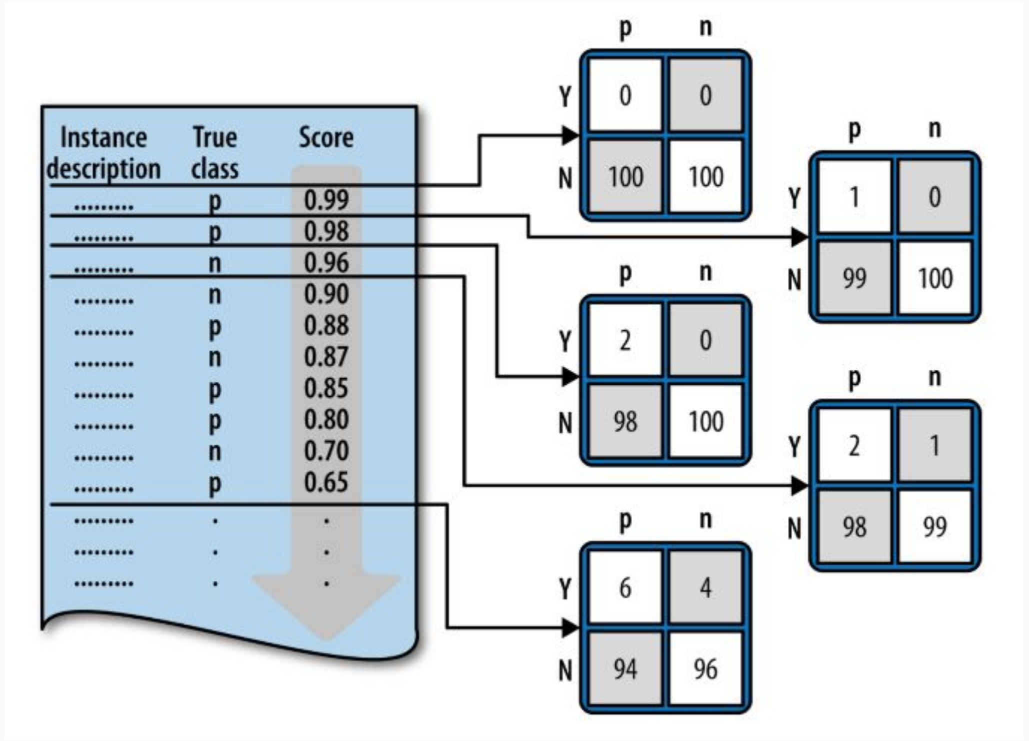


	p	n
Y	10	990
N	0	0

Cost-sensitive Learning - Effect of different thresholds

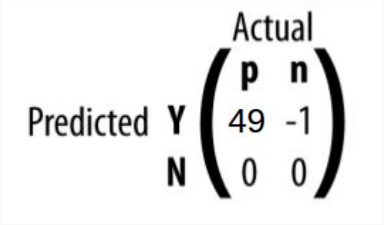
Find the profit-maximizing threshold

- Starting with the highest threshold (most probable) and working down compute expected profit.
- Then select threshold with highest expected profit (except if a budget is coming into play - next slide)
- Benefits of ranking (high prob. to low) clear when we have a budget and want to spend money on the most probable cases



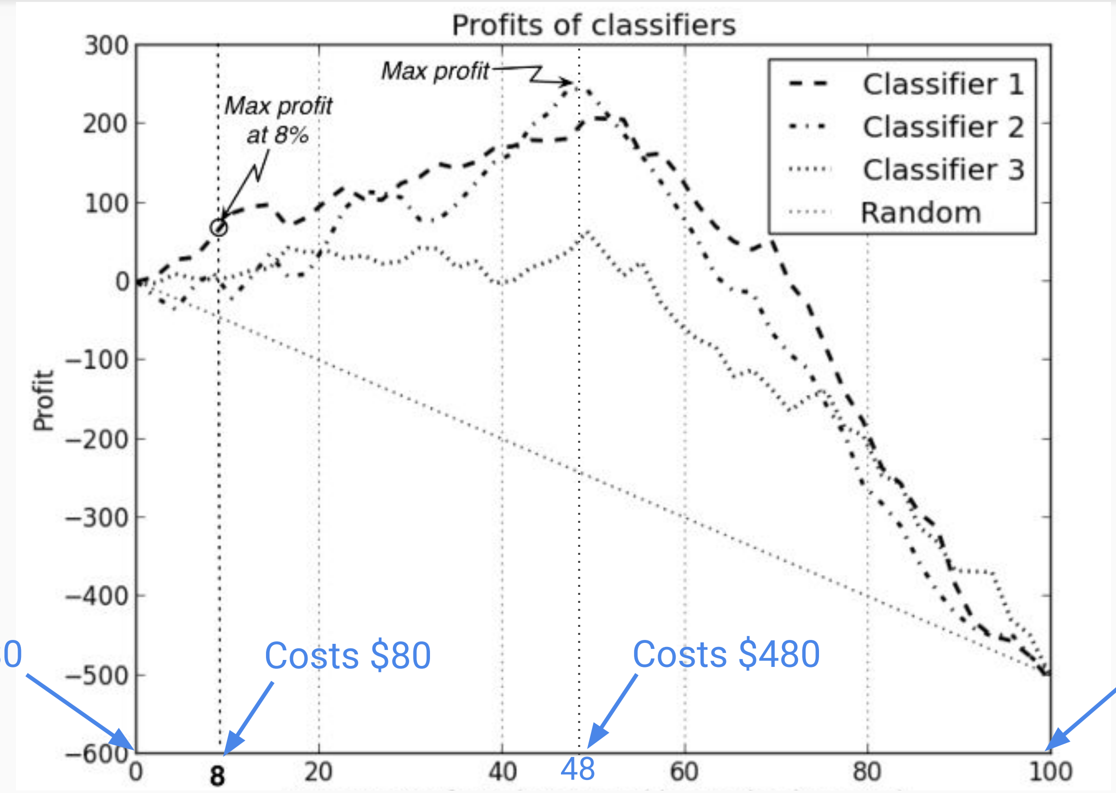
Profit Curve - budget could influence number of users you target

Cost - benefit matrix



$$\begin{array}{ccc} & & \text{Actual} \\ & p & n \\ \text{Predicted } Y & 49 & -1 \\ N & 0 & 0 \end{array}$$

Note no profit / cost on this matrix for FN, TN



Percent of test instances (decreasing by score)

Say there are 1000 instances. It costs $1 to check an instance.

Cost-benefit matrix - pitfalls

Correct

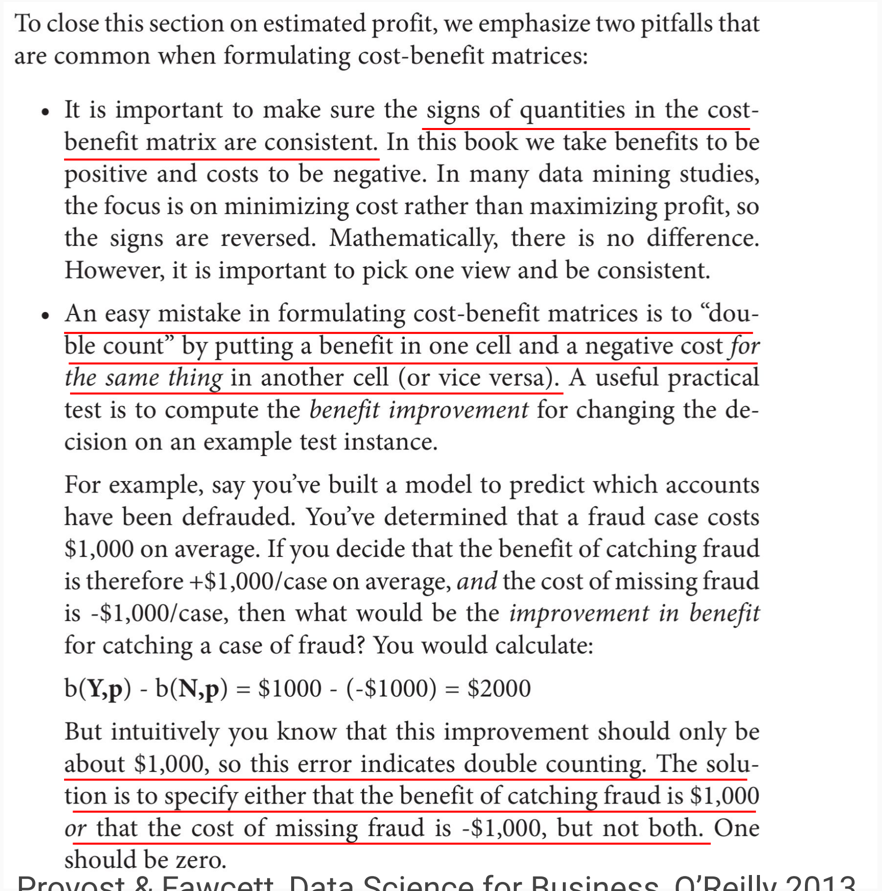
$$\begin{array}{ccc} & & \text{Actual} \\ & & \begin{array}{cc} \mathbf{p} & \mathbf{n} \end{array} \\ \text{Predicted} & \begin{array}{c} \mathbf{Y} \\ \mathbf{N} \end{array} & \begin{pmatrix} 49 & -1 \\ 0 & 0 \end{pmatrix} \end{array}$$

Incorrect

$$\begin{array}{ccc} & & \text{Actual} \\ & & \begin{array}{cc} \mathbf{p} & \mathbf{n} \end{array} \\ \text{Predicted} & \begin{array}{c} \mathbf{Y} \\ \mathbf{N} \end{array} & \begin{pmatrix} 49 & -1 \\ -50 & 0 \end{pmatrix} \end{array}$$

Don't double count!

Think about it all relative to the status quo.



To close this section on estimated profit, we emphasize two pitfalls that are common when formulating cost-benefit matrices:

- It is important to make sure the signs of quantities in the cost-benefit matrix are consistent. In this book we take benefits to be positive and costs to be negative. In many data mining studies, the focus is on minimizing cost rather than maximizing profit, so the signs are reversed. Mathematically, there is no difference. However, it is important to pick one view and be consistent.
- An easy mistake in formulating cost-benefit matrices is to "double count" by putting a benefit in one cell and a negative cost *for the same thing* in another cell (or vice versa). A useful practical test is to compute the *benefit improvement* for changing the decision on an example test instance.

  For example, say you've built a model to predict which accounts have been defrauded. You've determined that a fraud case costs \$1,000 on average. If you decide that the benefit of catching fraud is therefore +\$1,000/case on average, *and* the cost of missing fraud is -\$1,000/case, then what would be the *improvement in benefit* for catching a case of fraud? You would calculate:

  b(**Y,p**) - b(**N,p**) = \$1000 - (-\$1000) = \$2000

  But intuitively you know that this improvement should only be about \$1,000, so this error indicates double counting. The solution is to specify either that the benefit of catching fraud is \$1,000 *or* that the cost of missing fraud is -\$1,000, but not both. One should be zero.

Provost & Fawcett, Data Science for Business, O'Reilly 2013

Profit Curve Pseudocode

```
function profit_curve(cost_benefit, predicted_probs, labels):
    Sort instances by their prediction strength (the probabilities)
    For every instance in decreasing order of probability:
        Set the threshold to be the probability
        Set everything above the threshold to the positive class
        Calculate the confusion matrix
        element-wise multiply the confusion matrix by the cost_benefit
    Return an array of the profits and their associated thresholds
```

Check Understanding

Create a cost-benefit matrix for the following scenarios. Assume in each case that the status quo is no early detection/action

- A credit card company is running a fraud detection algorithm. On average, it costs $10 to investigate possible fraudulent charges, and $250 for each fraudulent charge that goes through.

- A ride-sharing company tries to predict when customers will churn (stop using the service), and attempts to retain the customer. They deploy a $25 coupon to try and retain those customers, whose lost business would cost them $150 over the next year.

Check Understanding

In each of the scenarios, compute the profit or loss for the following confusion matrix:

	p	n
Y	10	190
N	100	700

- A credit card company is running a fraud detection algorithm. On average, it costs $10 to investigate possible fraudulent charges, and $250 for each fraudulent charge that goes through.

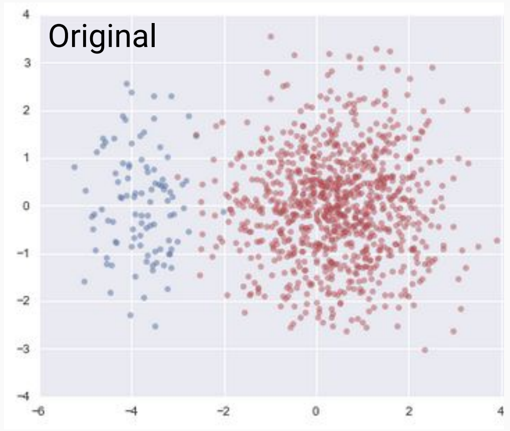
- A ride-sharing company tries to predict when customers will churn (stop using the service), and attempts to retain the customer. They deploy a $25 coupon to try and retain those customers, whose lost business would cost them $150 over the next year.

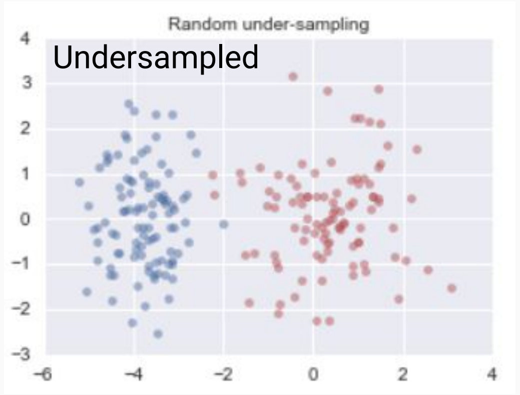
Dealing with imbalanced classes : Sampling techniques

galvanize

Sampling Techniques - Undersampling

- Undersampling randomly discards majority class observations to balance training sample.
- **PRO:** Reduces runtime on very large datasets.
- **CON:** Discards potentially important observations. Can overfit





https://github.com/scikit-learn-contrib/imbalanced-learnc

Mention imbalanced-learn package (https://github.com/scikit-learn-contrib/imbalanced-learn)

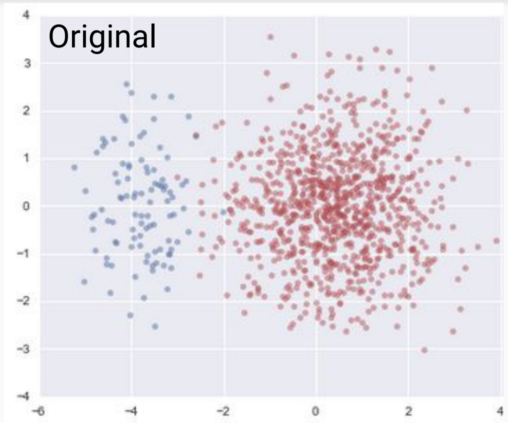
Sampling Techniques - Oversampling

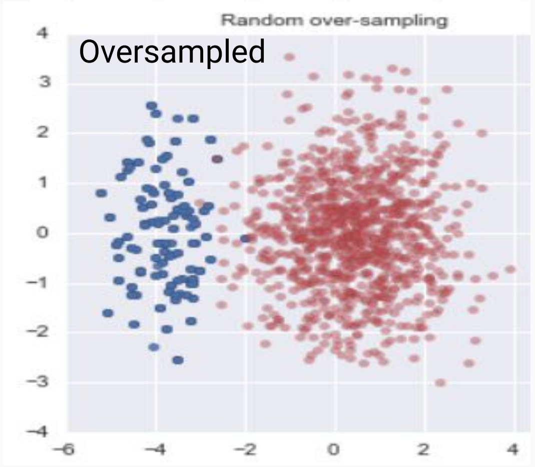
- Oversampling replicates observations from minority class to balance training sample.

- **PRO:** Doesn't discard information.

- **CON:** Likely to overfit to oversampled class.

(Often better to use SMOTE)





https://github.com/scikit-learn-contrib/imbalanced-learnc

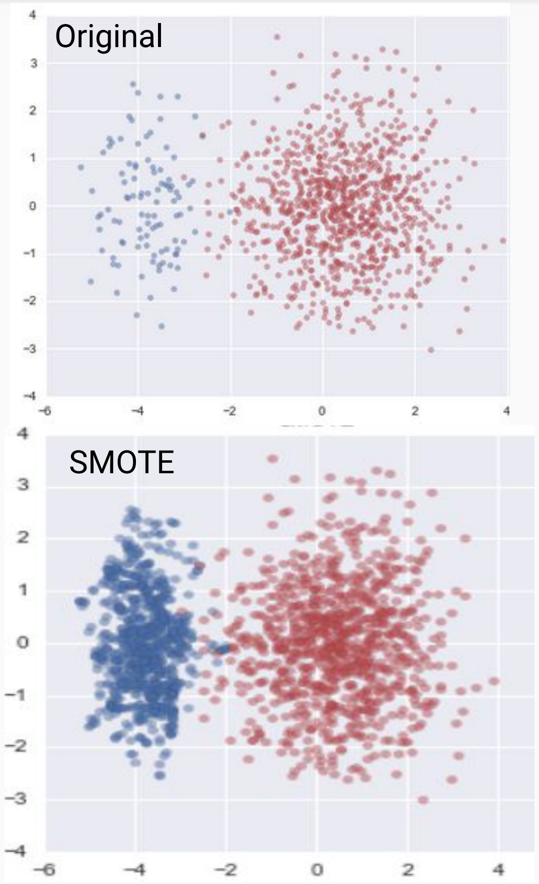
Sampling Techniques - SMOTE

galvanize

- SMOTE - Synthetic Minority Oversampling TechniquE

- Generates new observations from minority class.

- For each minority class observation and for each feature, randomly generate between it and one of its k-nearest neighbors.



https://github.com/scikit-learn-contrib/imbalanced-learnc

Sampling Techniques - SMOTE

SMOTE pseudocode

# of desired minority observations

```
synthetic_observations = []
while len(synthetic_observations) + len(minority_observations) < target:
    obs = random.choice(minority_observations)
    neighbor = random.choice(kNN(obs, k)) # randomly selected neighbor
    new_observation = {}
    for feature in obs:
        weight = random() # random float between 0 and 1
        new_feature_value = weight*obs[feature] + (1-weight)*neighbor[feature]
        new_observation[feature] = new_feature_value
    synthetic_observations.append(new_observation)
```

See also psuedo-code in original SMOTE paper:
https://www.jair.org/media/953/live-953-2037-jair.pdf

Sampling Techniques - Distribution

- **What's the right amount of over-/under-sampling?**
- If you know the cost-benefit matrix:
  - Maximize profit curve over target proportion
- If you don't know the cost-benefit matrix:
  - No clear answer...
  - AUROC's might be more useful

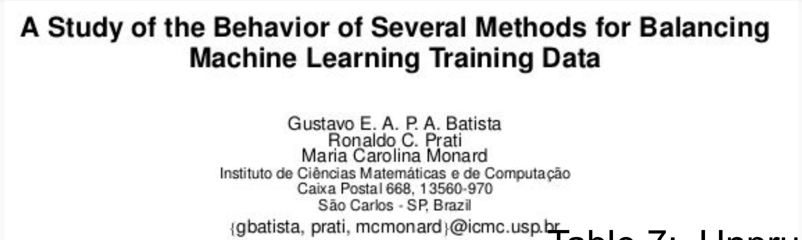
Best option: Can you get more data???

Cost Sensitivity vs Sampling

- Neither is strictly superior.
- Oversampling tends to work better than undersampling on small datasets.
- Some algorithms don’t have an obvious cost-sensitive adaptation, requiring sampling modifications

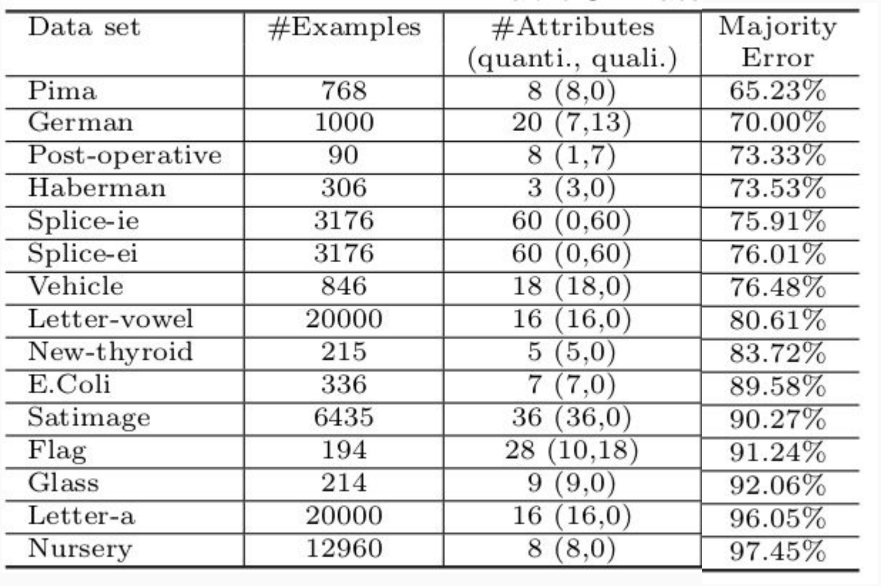
See also “Cost-Sensitive Learning vs. Sampling: Which is Best for Handling Unbalanced Classes with Unequal Error Costs?”
http://storm.cis.fordham.edu/gweiss/papers/dmin07-weiss.pdf

What's "best?" No panacea for all cases



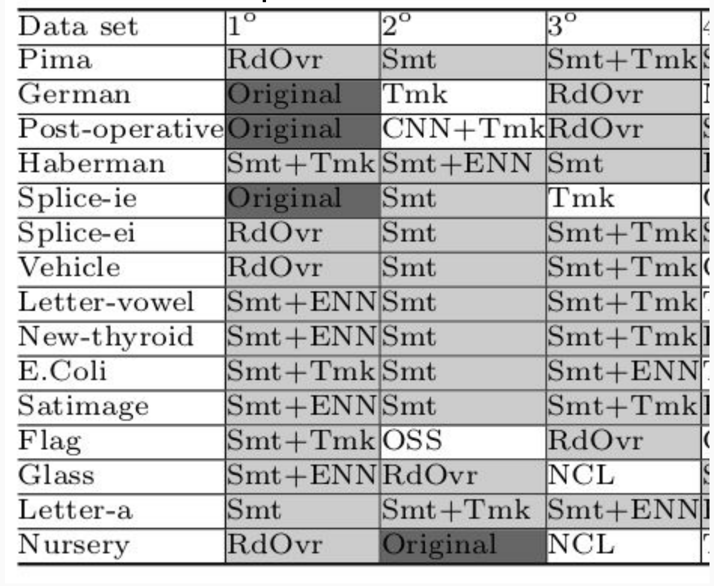
A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data

Gustavo E. A. P. A. Batista
Ronaldo C. Prati
Maria Carolina Monard
Instituto de Ciências Matemáticas e de Computação
Caixa Postal 668, 13560-970
São Carlos - SP, Brazil
{gbatista, prati, mcmonard}@icmc.usp.br



Data set	#Examples	#Attributes (quanti., quali.)	Majority Error
Pima	768	8 (8,0)	65.23%
German	1000	20 (7,13)	70.00%
Post-operative	90	8 (1,7)	73.33%
Haberman	306	3 (3,0)	73.53%
Splice-ie	3176	60 (0,60)	75.91%
Splice-ei	3176	60 (0,60)	76.01%
Vehicle	846	18 (18,0)	76.48%
Letter-vowel	20000	16 (16,0)	80.61%
New-thyroid	215	5 (5,0)	83.72%
E.Coli	336	7 (7,0)	89.58%
Satimage	6435	36 (36,0)	90.27%
Flag	194	28 (10,18)	91.24%
Glass	214	9 (9,0)	92.06%
Letter-a	20000	16 (16,0)	96.05%
Nursery	12960	8 (8,0)	97.45%

Table 7: Unpruned decision trees



Data set	1°	2°	3°	4
Pima	RdOvr	Smt	Smt+Tmk	S
German	Original	Tmk	RdOvr	I
Post-operative	Original	CNN+Tmk	RdOvr	S
Haberman	Smt+Tmk	Smt+ENN	Smt	I
Splice-ie	Original	Smt	Tmk	(
Splice-ei	RdOvr	Smt	Smt+Tmk	S
Vehicle	RdOvr	Smt	Smt+Tmk	(
Letter-vowel	Smt+ENN	Smt	Smt+Tmk	T
New-thyroid	Smt+ENN	Smt	Smt+Tmk	I
E.Coli	Smt+Tmk	Smt	Smt+ENN	T
Satimage	Smt+ENN	Smt	Smt+Tmk	I
Flag	Smt+Tmk	OSS	RdOvr	(
Glass	Smt+ENN	RdOvr	NCL	S
Letter-a	Smt	Smt+Tmk	Smt+ENN	I
Nursery	RdOvr	Original	NCL	T

Review Questions

- What are some problems with using accuracy to evaluate models?
- What are some better metrics than accuracy?
- What are the three ways to deal with imbalanced classes?
- Which way would you recommend if you know something about the business costs of false positives and false negatives?

Recap: Learning Objectives

✓ **Review:** confusion matrices and classification metrics

✓ **Discuss** challenges with imbalanced datasets

✓ **Learn, compare, and be ready to implement** 3 methods of dealing with imbalanced data:
- Practical steps (tweak your model)
- Cost sensitive learning (consider costs of FP, FN)
- Sampling methods (balance your classes)