

Profit Curves and Imbalanced Classes

Frank Burkholder
Ryan Henning



After this lecture you should be able to:

- Explain what an imbalanced data set is
- Choose appropriate metrics to quantify results in an imbalanced data set.
- Propose several different approaches to getting good results out of an imbalanced data set
- Explain what double counting is in a cost-benefit matrix, and avoid it
- Construct a profit curve

“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?

- 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.*
- The cost (in time, money, etc.) of a false positive is often different from the 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.

Practical steps (help your model fit better):

- Stratifying train_test_split
- Change minority & majority weighting of training data

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 - Synthetic Minority Oversampling TEchnique

Dealing with imbalanced classes: Practical steps

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

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)
```

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)
```

What if very few of the minority class end up in `y_test`!?

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)
```

What if very few of the minority class end up in `y_test`!?

This is better:

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

Practical steps - change point weighting

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) ¶
```

[\[source\]](#)

class_weight : dict or 'balanced', optional

Option 1

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.

```
In [21]: n_classes = 2
In [22]: n1 = 10
In [23]: n2 = 90
In [24]: n_samples = n1 + n2
In [25]: w1 = n_samples / (n_classes * n1)
In [26]: w2 = n_samples / (n_classes * n2)
In [27]: print(f"w1: {w1:0.2f}, w2: {w2:0.2f}.")
w1: 5.00, w2: 0.56.
```

Option 2

fit (X, y[, sample_weight])

Fit the model according to the given training data.

Dealing with imbalanced classes: Cost sensitive learning (Profit Curves)

- 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 (subject to budget constraints).

Cost-sensitive learning: find threshold at max profit

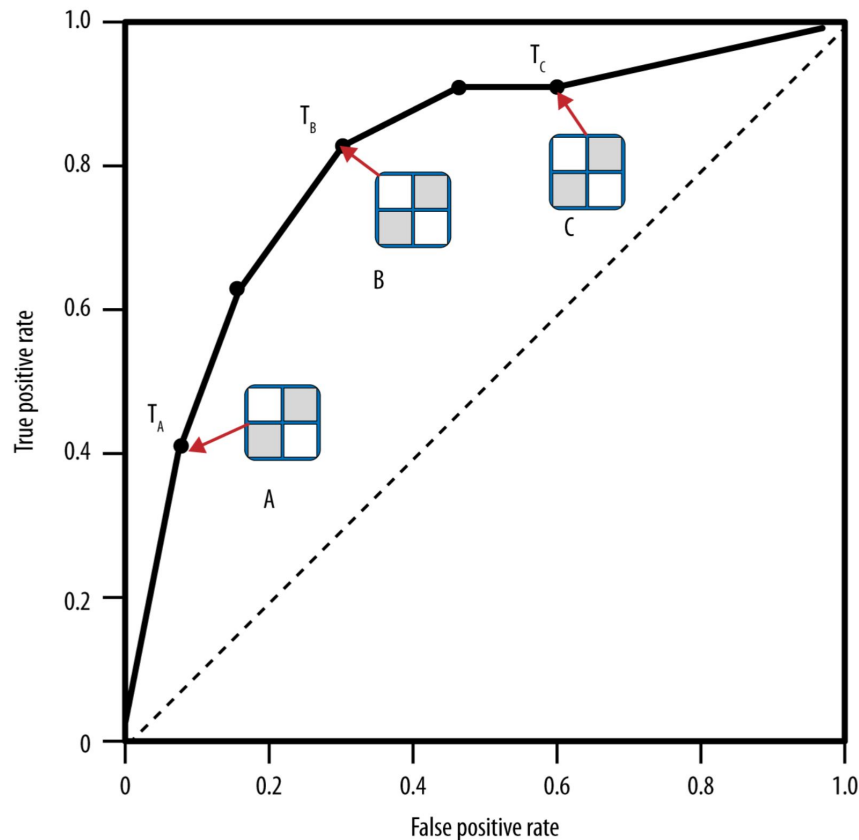
Recall the ROC Curve:

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

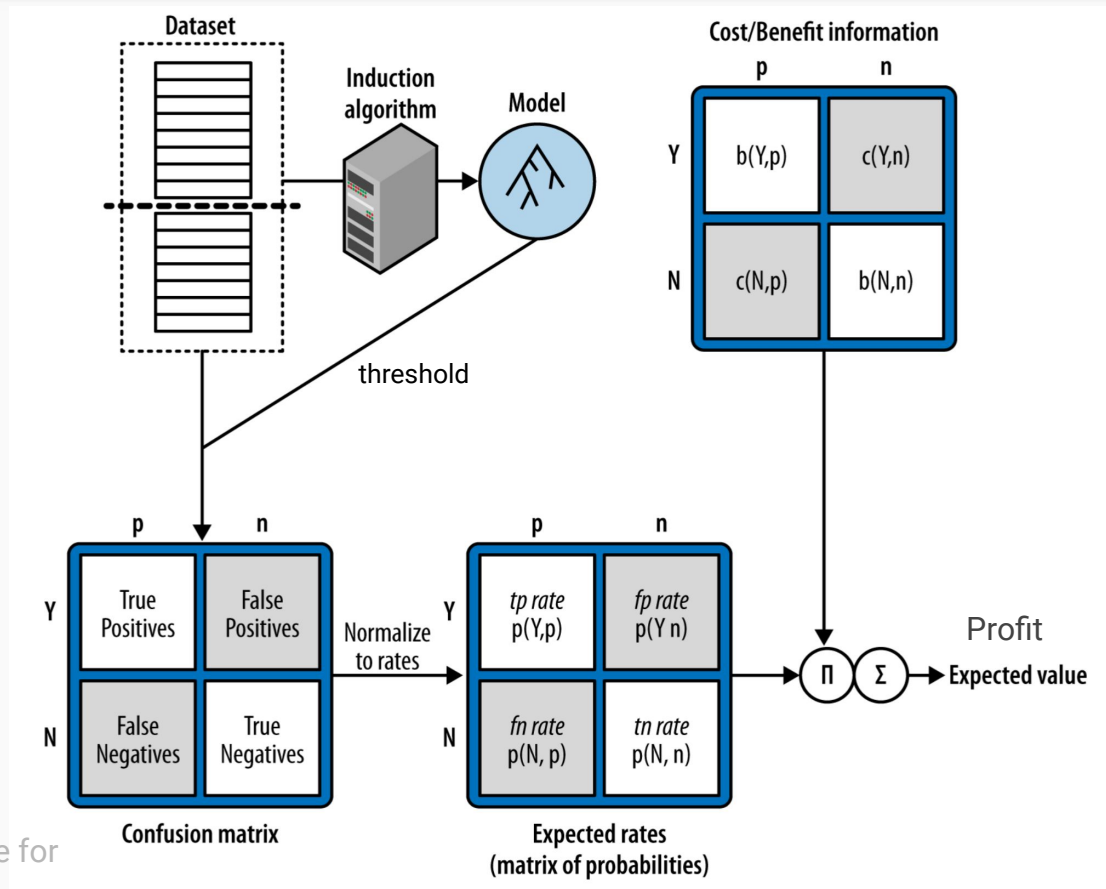
Q1: How to handle unequal error costs?

Q2: Which threshold to pick?

A1&2: Plot expected profit (4 steps)!

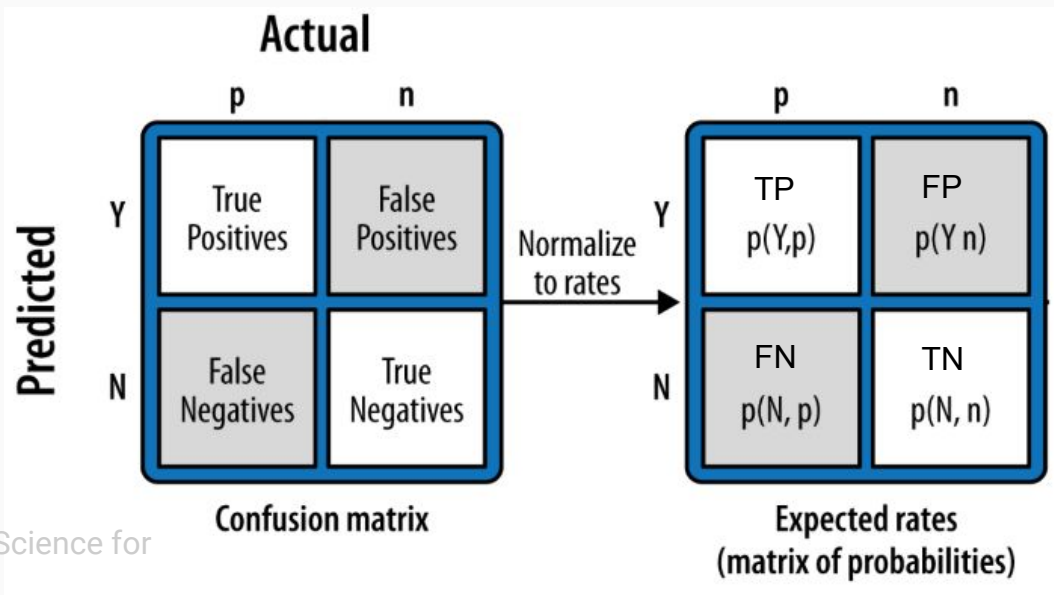


Cost-sensitive learning - Overview



Computing Expected Profit

Step 1 - Estimate probabilities based on a given threshold and a confusion matrix



Rates are counts in cell divided by **total of all counts in matrix.**

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)

		Actual	
		p	n
Predicted	Y	49	-1
	N	0	0

\$50 - \$1

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

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

Correct

		Actual	
		p	n
Predicted	Y	49	-1
	N	0	0

Incorrect (double counting)

		Actual	
		p	n
Predicted	Y	49	-1
	N	-50	0

Think about it 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.

Example of a cost-benefit matrix

Let's say a fraudulent transaction costs us **\$1000**.

Let's say investigating a transaction on suspicion of fraud costs us **\$100**.

What does our cost benefit matrix look like?

		Actual	
		Fraud (p)	Not Fraud (n)
Predict	Fraud (Y)		
	Not F. (N)		

Example of a cost-benefit matrix

Let's say a fraudulent transaction costs us **\$1000**.

Let's say investigating a transaction on suspicion of fraud costs us **\$100**.

What does our cost benefit matrix look like?

It's all about perspective.

		Actual	
		Fraud (p)	Not Fraud (n)
Predict	Fraud (Y)		
	Not F. (N)		

Example of a cost-benefit matrix

Let's say a fraudulent transaction costs us **\$1000**.

Let's say investigating a transaction on suspicion of fraud costs us **\$100**.

What does our cost benefit matrix look like?

It's all about perspective.

A budget that doesn't include costs (nothing is budgeted for fraud or intervening to try to prevent it.)

So if a transaction is not fraudulent, and we don't pay to investigate it, we are on budget (\$0).

		Actual	
		Fraud (p)	Not Fraud (n)
Predict	Fraud (Y)		
	Not F. (N)		\$0

Example of a cost-benefit matrix

Let's say a fraudulent transaction costs us **\$1000**.

Let's say investigating a transaction on suspicion of fraud costs us **\$100**.

What does our cost benefit matrix look like?

It's all about perspective.

A budget that doesn't include costs (nothing is budgeted for fraud or intervening to try to prevent it.)

So if a transaction is not fraudulent, and we don't pay to investigate it, we are on budget (\$0).

But if we suspect fraud, we investigate (-\$100) and if it was fraud, prevent it.

		Actual	
		Fraud (p)	Not Fraud (n)
Predict	Fraud (Y)	-\$100	-\$100
	Not F. (N)		\$0

Example of a cost-benefit matrix

Let's say a fraudulent transaction costs us **\$1000**.

Let's say investigating a transaction on suspicion of fraud costs us **\$100**.

What does our cost benefit matrix look like?

It's all about perspective.

A budget that doesn't include costs (nothing is budgeted for fraud or intervening to try to prevent it.)

So if a transaction is not fraudulent, and we don't pay to investigate it, we are on budget (\$0).

But if we suspect fraud, we investigate (-\$100) and if it was fraud, prevent it.

But if it was fraud, and we didn't investigate it, it costs us. (-\$1000)

		Actual	
		Fraud (p)	Not Fraud (n)
Predict	Fraud (Y)	-\$100	-\$100
	Not F. (N)	-\$1000	\$0

Example of a cost-benefit matrix

Let's say a fraudulent transaction costs us **\$1000**.

Let's say investigating a transaction on suspicion of fraud costs us **\$100**.

What does our cost benefit matrix look like?

It's all about perspective.

A budget that doesn't include costs (nothing is budgeted for fraud or intervening to try to prevent it.)

So if a transaction is not fraudulent, and we don't pay to investigate it, we are on budget (\$0).

But if we suspect fraud, we investigate (-\$100) and if it was fraud, prevent it.

But if it was fraud, and we didn't investigate it, it costs us. (-\$1000)

		Actual	
		Fraud (p)	Not Fraud (n)
Predict	Fraud (Y)	-\$100	-\$100
	Not F. (N)	-\$1000	\$0

This cb matrix lends itself to a minimizing cost, rather than maximizing profit, perspective.

Breakout - pair up

Let's say a fraudulent transaction costs us \$1000.

Let's say investigating a transaction on suspicion of fraud costs us \$100.

What does our cost benefit matrix look like?

It's all about perspective.

A budget that assumes current cases of fraud are the status quo (so if it is fraud and we don't investigate: \$0), and we don't budget for investigating fraud.

What does this cost benefit matrix look like?

		Actual	
		Fraud (p)	Not Fraud (n)
Predict	Fraud (Y)		
	Not F. (N)		

Breakout - solution

Let's say a fraudulent transaction costs us \$1000.

Let's say investigating a transaction on suspicion of fraud costs us \$100.

What does our cost benefit matrix look like?

It's all about perspective.

A budget that assumes current cases of fraud are the status quo (so if it is fraud and we don't investigate: \$0), and we don't budget for investigating fraud.

What does this cost benefit matrix look like?

		Actual	
		Fraud (p)	Not Fraud (n)
Predict	Fraud (Y)	\$900	-\$100
	Not F. (N)	\$0	\$0

This cb matrix lends itself to maximizing profit, rather than minimizing cost.

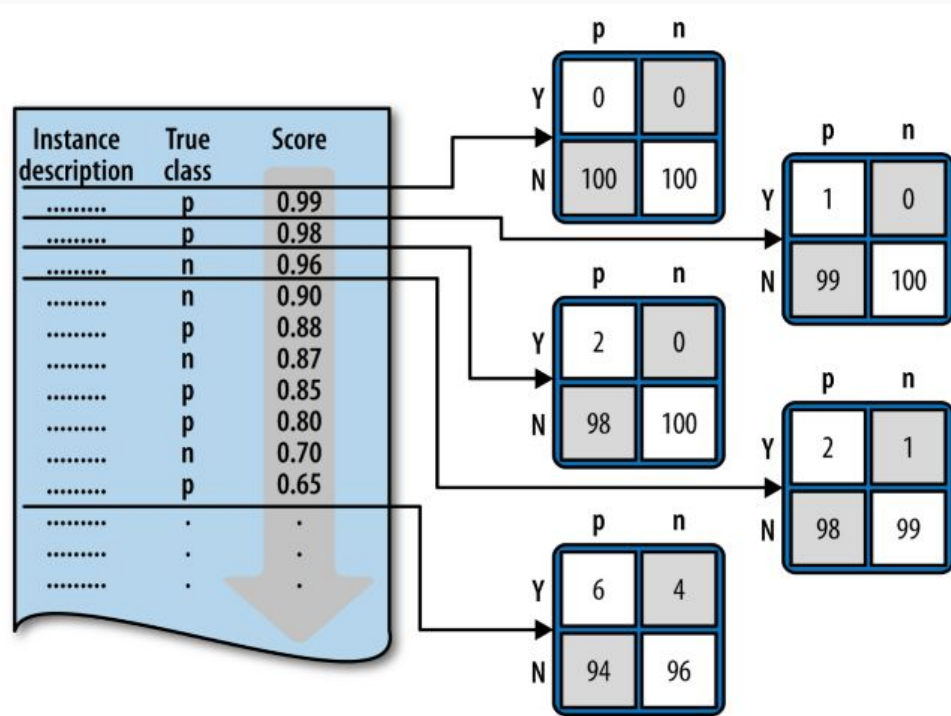
Computing Expected Profit

Step 3 - Combine probabilities and payoffs.

$$E[Profit] = \begin{matrix} & \text{TP} & \text{FP} \\ P(Y, p) \cdot b(Y, p) + P(Y, n) \cdot c(Y, n) + \\ & \text{FN} & \text{TN} \\ P(N, p) \cdot c(N, p) + P(N, n) \cdot b(N, n) \end{matrix}$$

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



The threshold makes a difference!

Let me show you:

```
the_classification_threshold_makes_a_difference.ipynb
```

Cost-sensitive learning: Plot profit

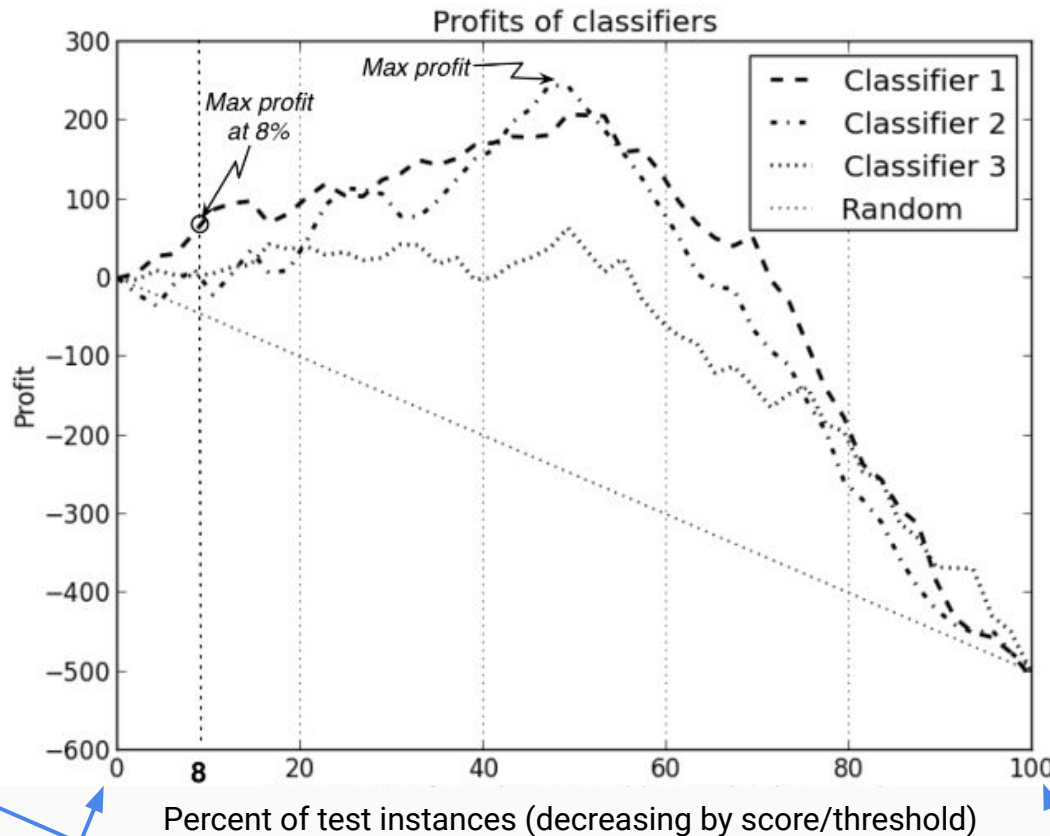
Step 4 - Plot profit

Cost - benefit matrix

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

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

	p	n
Y	0	0
N	10	990



	p	n
Y	10	990
N	0	0

Threshold = 1.0

<- Sorted by threshold ->

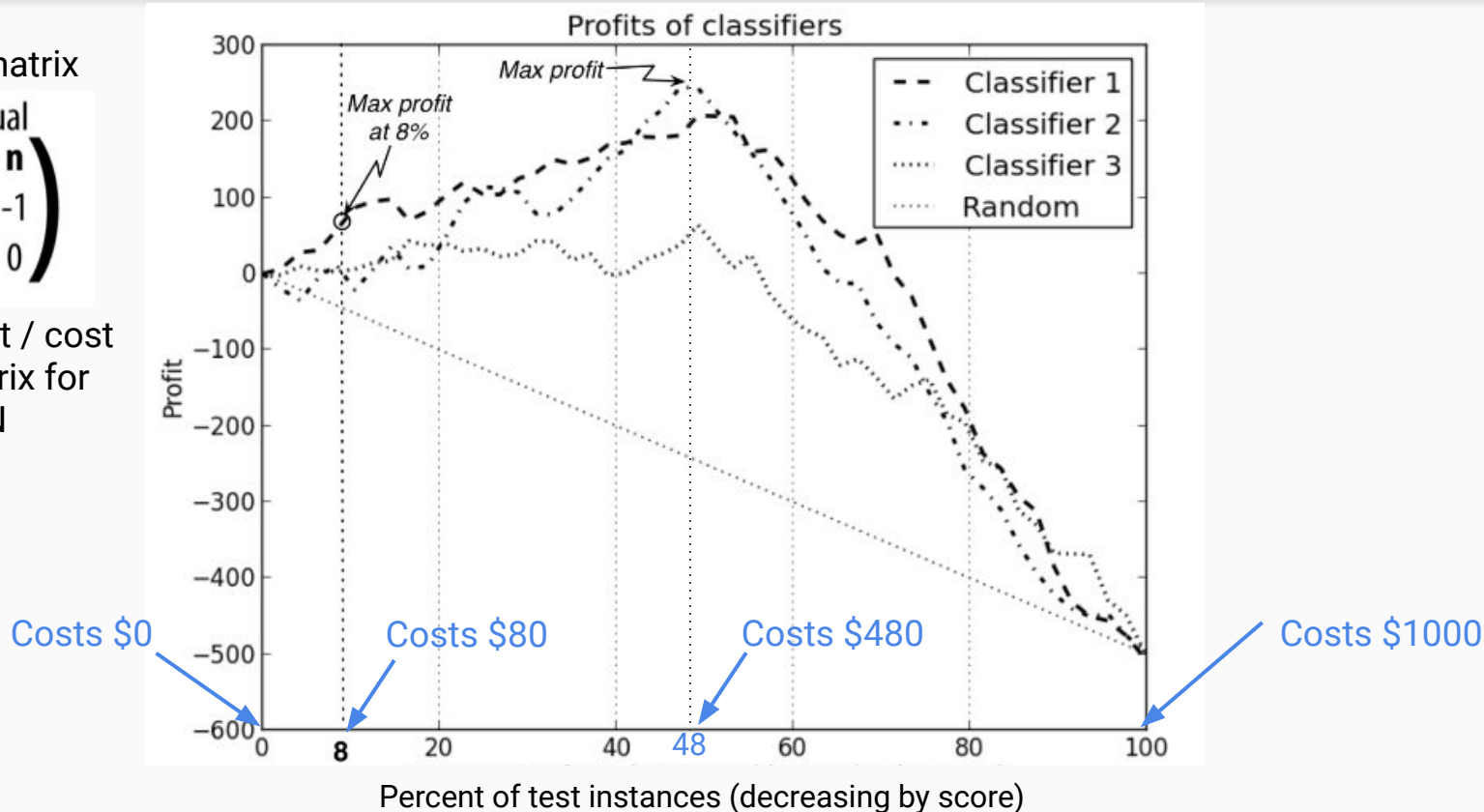
Threshold = 0.0

Profit Curve - budget affects those you target

Cost - benefit matrix

		Actual	
Predicted	Y	p	n
	N	49	-1
		0	0

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



Say there are 1000 instances (rows). It costs \$1 to check an instance.

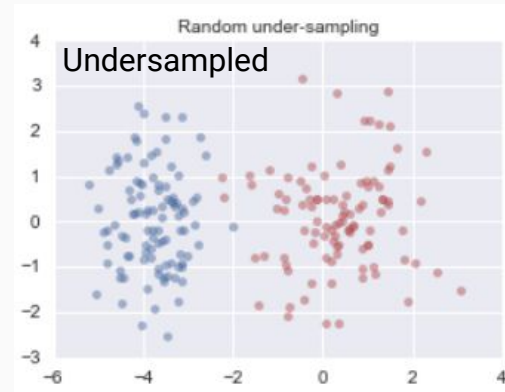
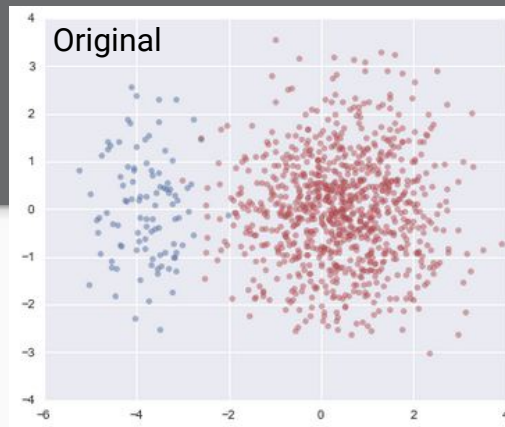
Dealing with imbalanced classes: Sampling techniques

- Undersampling
- Oversampling
- SMOTE - Synthetic Minority Oversampling Technique

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.

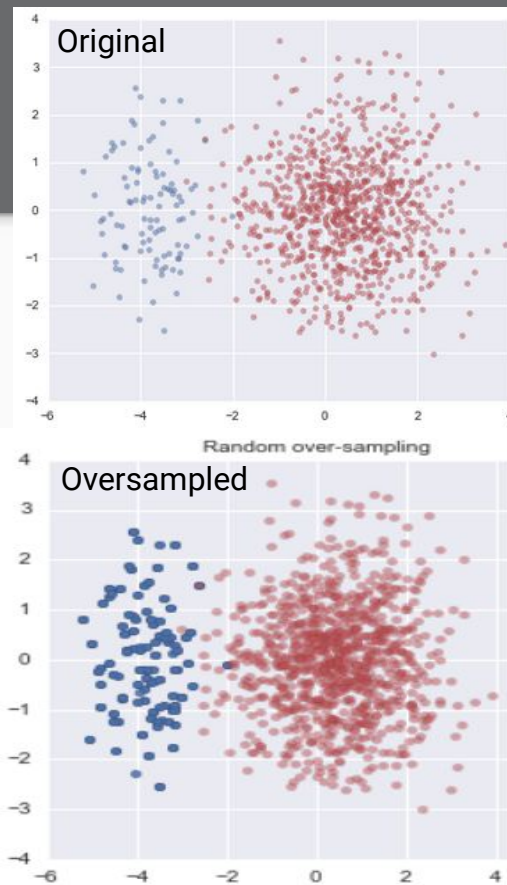
Mention imbalanced-learn package
(<https://github.com/scikit-learn-contrib/imbalanced-learn>)



Sampling Techniques - Oversampling

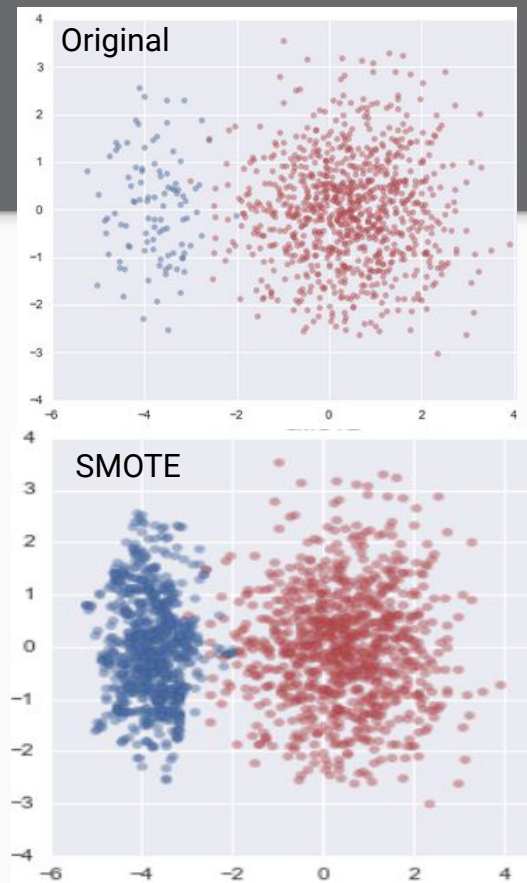
- Oversampling replicates observations from minority class to balance training sample.
- **PRO:** Doesn't discard information.
- **CON:** Likely to overfit.

(Often better to use SMOTE)



Sampling Techniques: SMOTE


- 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.



SMOTE pseudocode

```
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)
```

of desired
minority
observations



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...
 - ROC's AUC might be more useful...

Cross-validation and sampling techniques

Discuss:

Where in the cross-validation process do you apply sampling techniques?

A follow up question:

Let's say you had severe class imbalance. You applied SMOTE and got amazing performance during cross-validation. But then on your hold-out set you had abysmal performance. What do you think went wrong?

Cross-validation and sampling techniques

Discuss:

Where in the cross-validation process do you apply sampling techniques?

You apply it to your train set (not your test set, and not your hold-out set).

A follow up question:

In the SMOTE example, it's likely SMOTE was applied to the entire dataset after the hold-out set was made, and then the SMOTEed datapoints ended up in train and test. SMOTEed datapoints are similar to existing datapoints, so in this case it was like testing on the training set.

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.

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?"

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°
Pima	RdOvr	Smt	Smt+Tmk
German	Original	Tmk	RdOvr
Post-operative	Original	CNN+Tmk	RdOvr
Haberman	Smt+Tmk	Smt+ENN	Smt
Splice-ie	Original	Smt	Tmk
Splice-ei	RdOvr	Smt	Smt+Tmk
Vehicle	RdOvr	Smt	Smt+Tmk
Letter-vowel	Smt+ENN	Smt	Smt+Tmk
New-thyroid	Smt+ENN	Smt	Smt+Tmk
E.Coli	Smt+Tmk	Smt	Smt+ENN
Satimage	Smt+ENN	Smt	Smt+Tmk
Flag	Smt+Tmk	OSS	RdOvr
Glass	Smt+ENN	RdOvr	NCL
Letter-a	Smt	Smt+Tmk	Smt+ENN
Nursery	RdOvr	Original	NCL

Review

Practical steps:

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

Cost-sensitive learning:

- profit curves (maximize profit from cb matrix and threshold)

Sampling:

- Oversampling
- Undersampling
- SMOTE

Best: Can you get more data?!?

After this lecture you should be able to:

- Explain what an imbalanced data set is
- Choose appropriate metrics to quantify results in an imbalanced data set.
- Propose several different approaches to getting good results out of an imbalanced data set
- Explain what double counting is in a cost-benefit matrix, and avoid it
- Construct a profit curve