Profit Curves and Imbalanced Classes

Ryan Henning's slides
Customized by Frank Burkholder
9 May 2017

galvanize

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.

Dealing with imbalanced classes: Practical steps

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

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)
```

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 - Synthetic Minority Oversampling TEchnique

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)
```

Practical steps - weight points of minority class

galvanize

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.

Investigate benefit of stratification and class weights

Jupyter notebook

Dealing with imbalanced classes: Cost sensitive learning

- 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

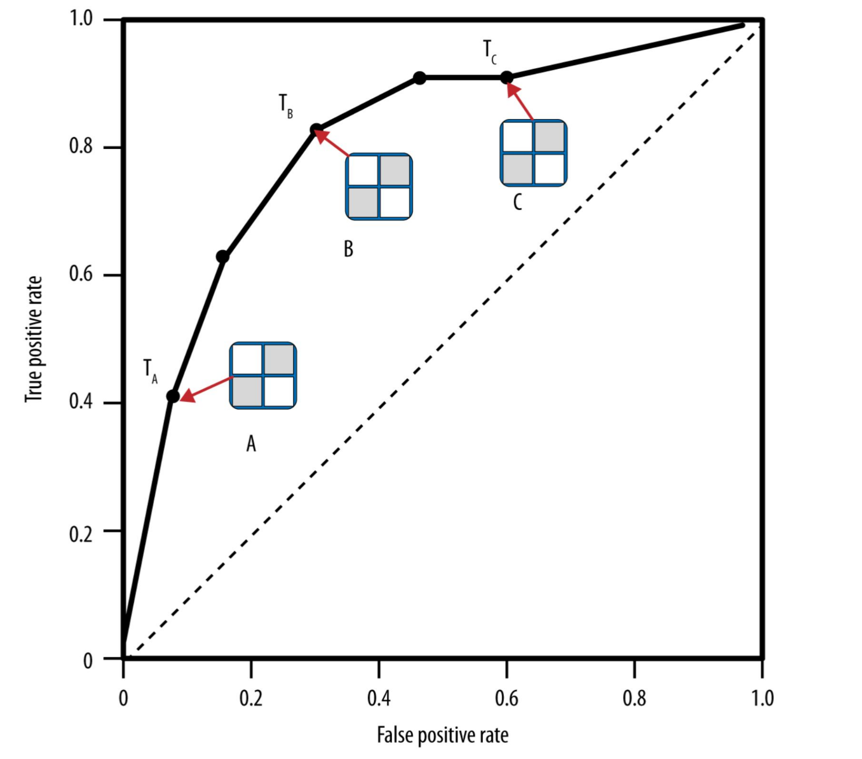
Recall the ROC Curve:

- 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?

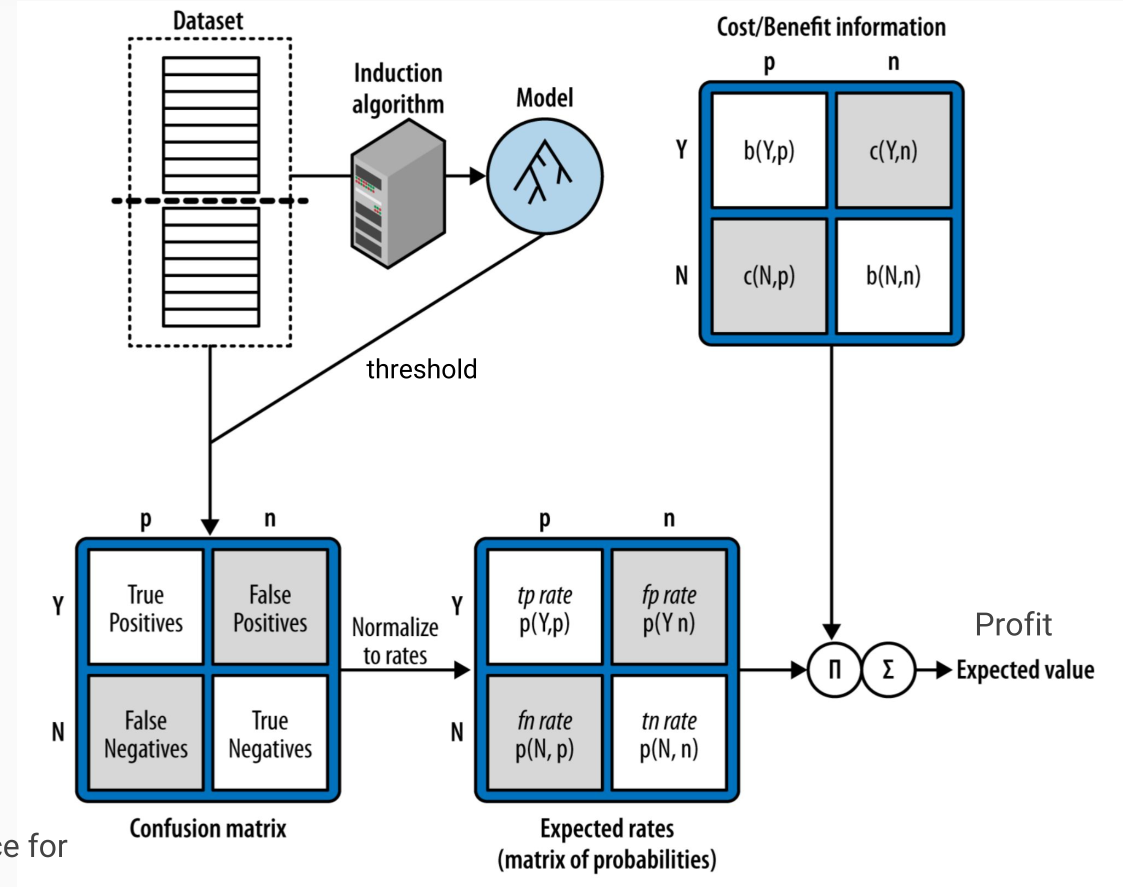
Q: Which threshold to pick? (assessment)

A: Plot expected profit (4 steps)!



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

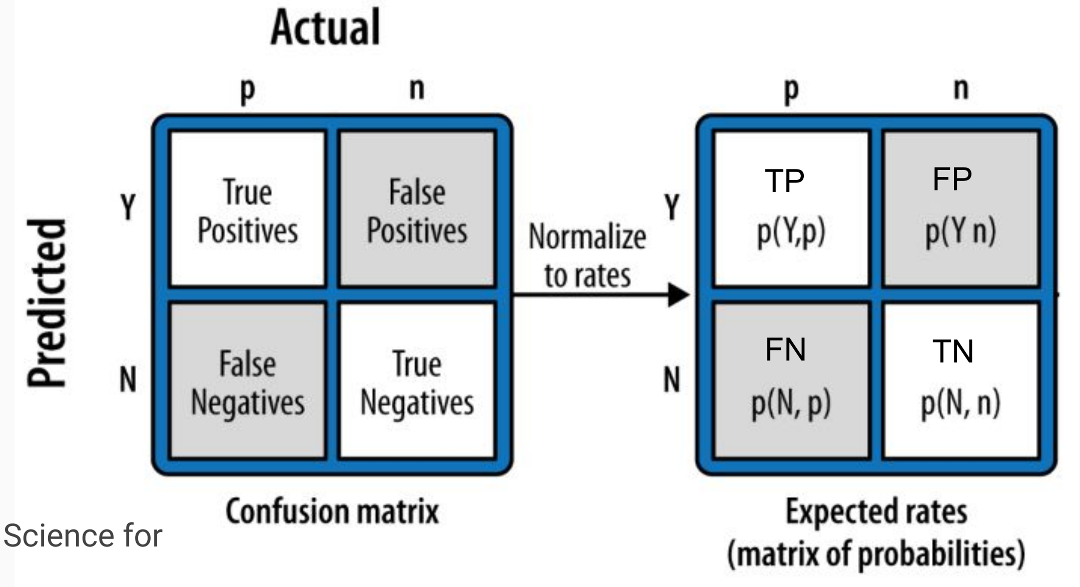
Cost-sensitive Learning - Overview



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

Computing Expected Profit

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



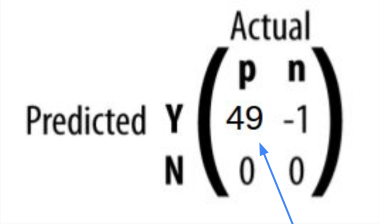
Provost & Fawcett, Data Science for Business, O'Reilly 2013

Cost-sensitive Learning - Thresholding

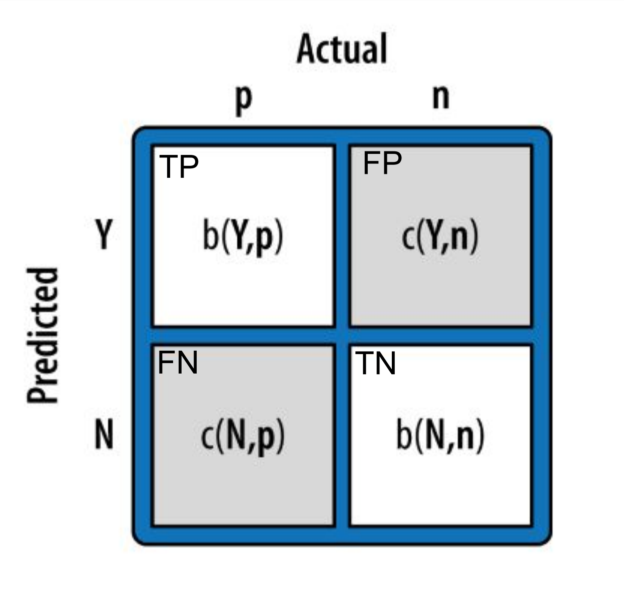
Computing Expected Profit

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

Example
(maybe filling out a survey)



$50 - $1



Cost-benefit matrix - pitfalls

galvanize

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:

  $b(\mathbf{Y},\mathbf{p}) - b(\mathbf{N},\mathbf{p}) = \$1000 - (-\$1000) = \$2000$

  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

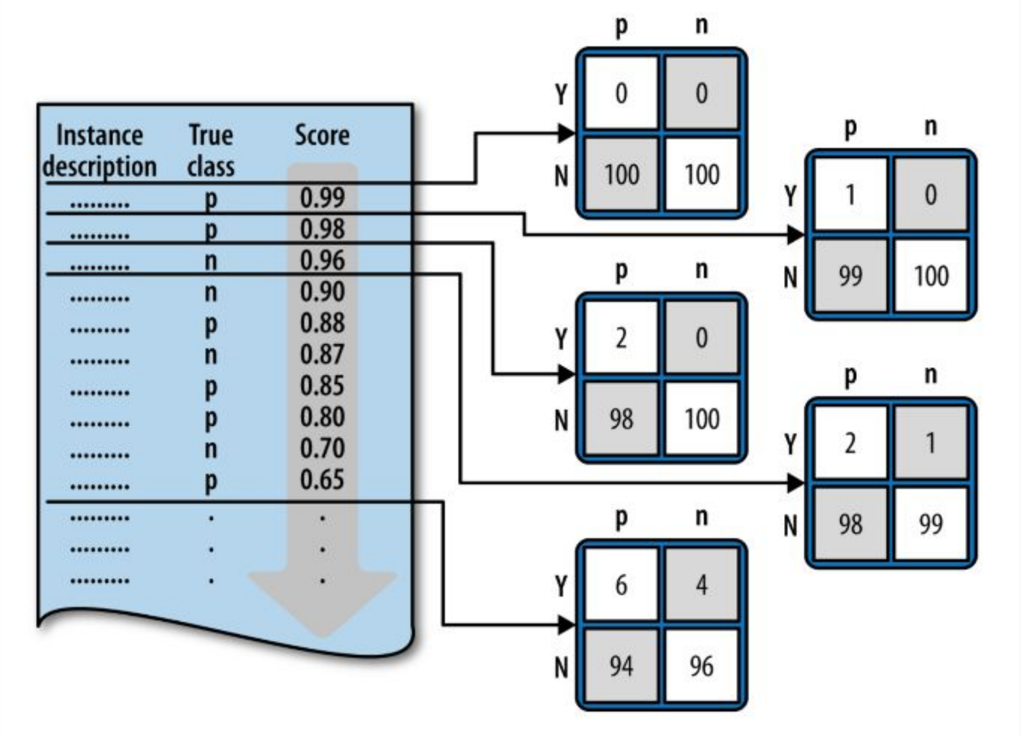
Computing Expected Profit

Step 3 - Combine probabilities and payoffs.

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

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



Cost-sensitive Learning - Thresholding and Profit Curve

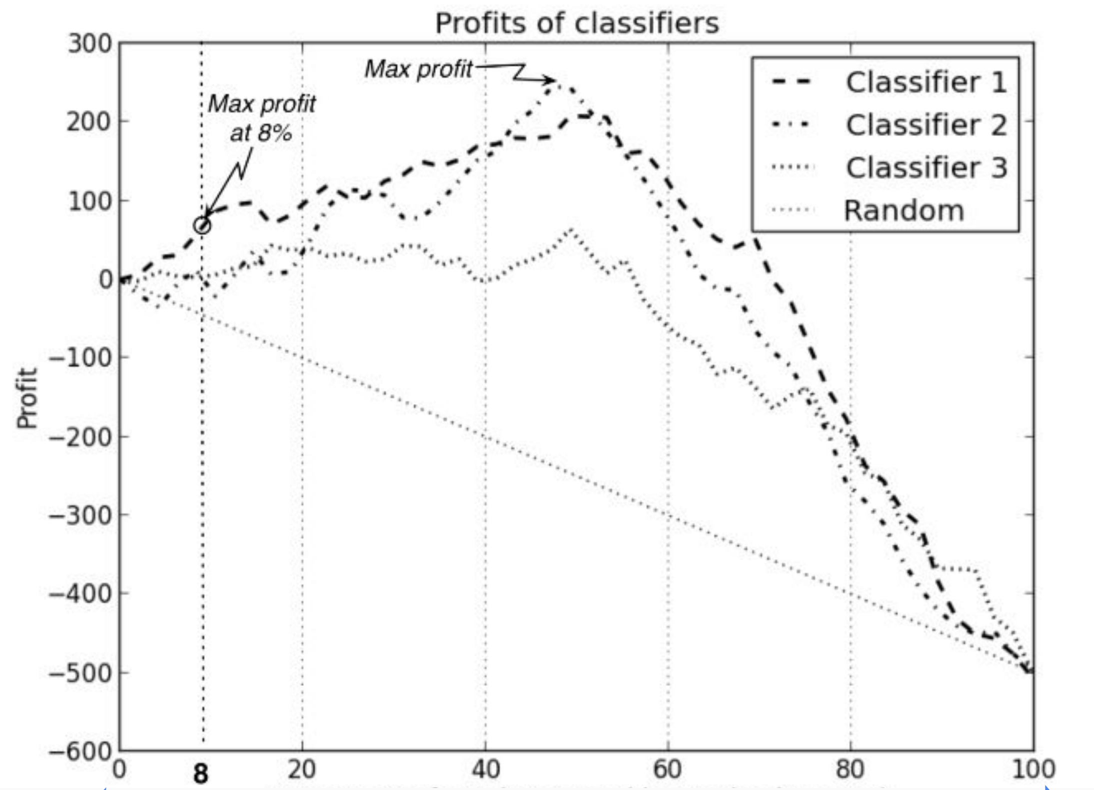
galvanize

Step 4 - Plot profit

Cost - benefit matrix

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

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 ->

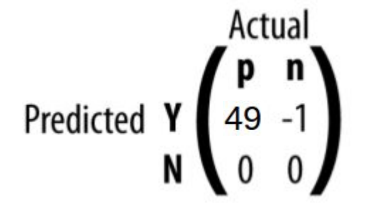
	p	n
Y	10	990
N	0	0

Threshold = 0.0

Profit Curve - budget could influence number of users you target

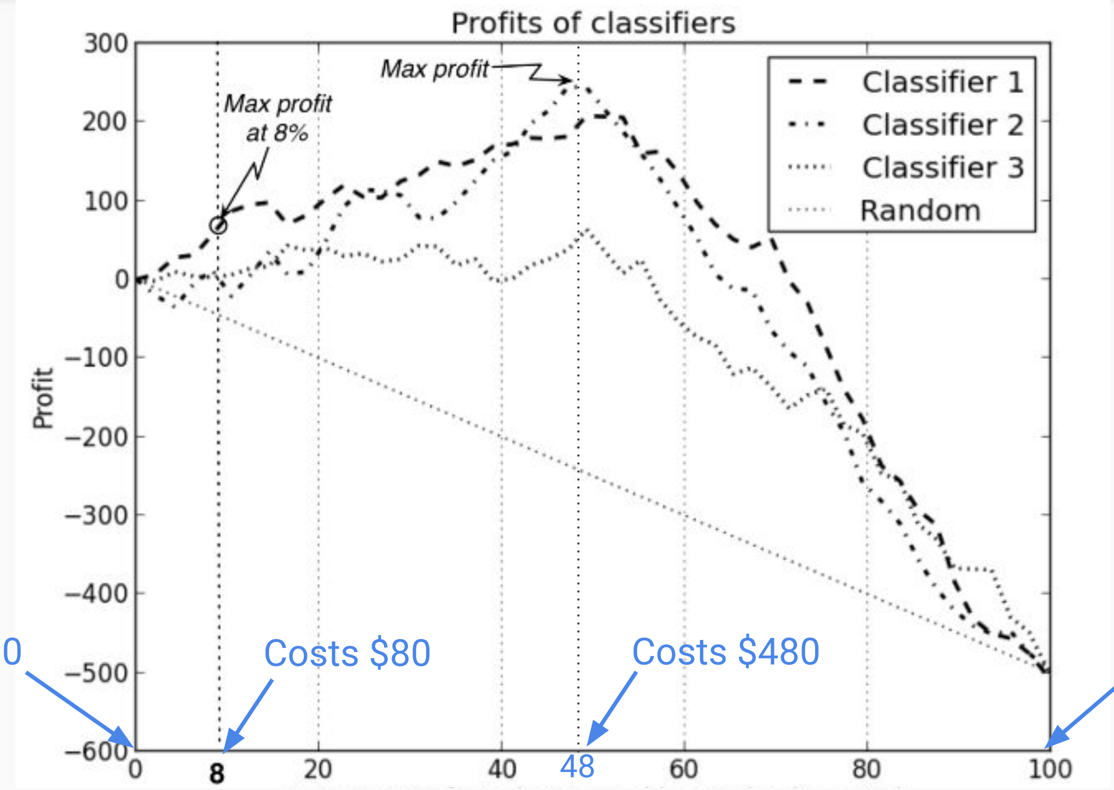
galvanize

Cost - benefit matrix



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

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.

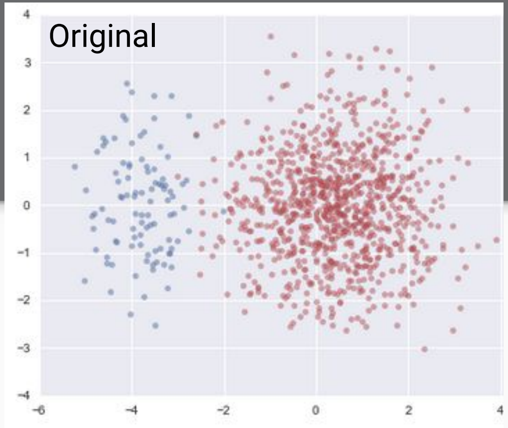
Dealing with imbalanced classes: Sampling techniques

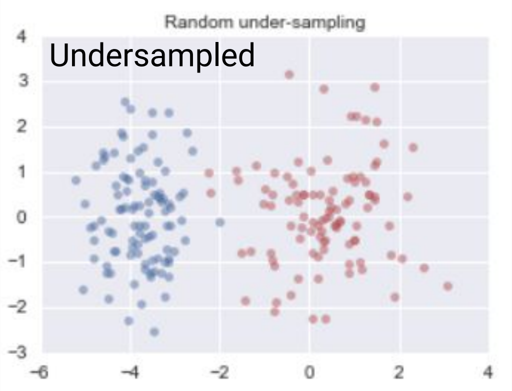
- Undersampling
- Oversampling
- SMOTE - Synthetic Minority Oversampling TechniquE

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)



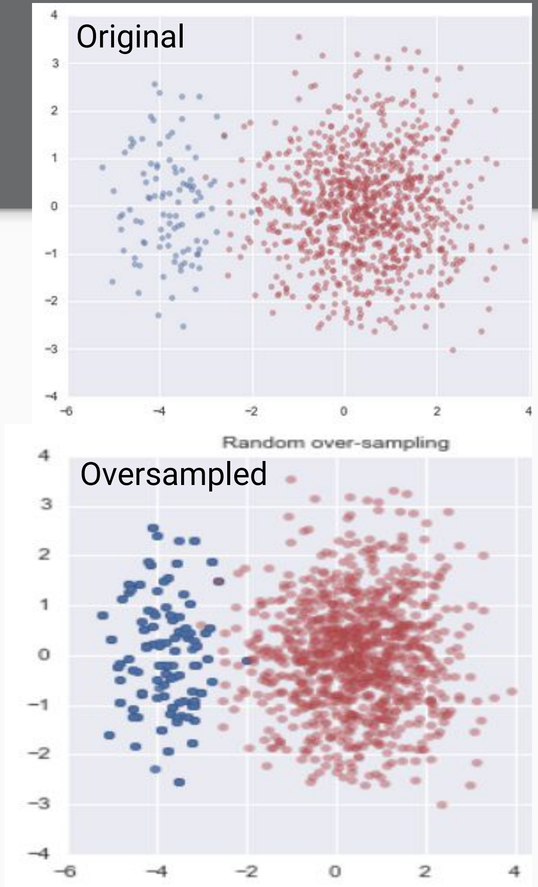


https://github.com/scikit-learn-contrib/imbalanced-learnc

Sampling Techniques - Oversampling

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

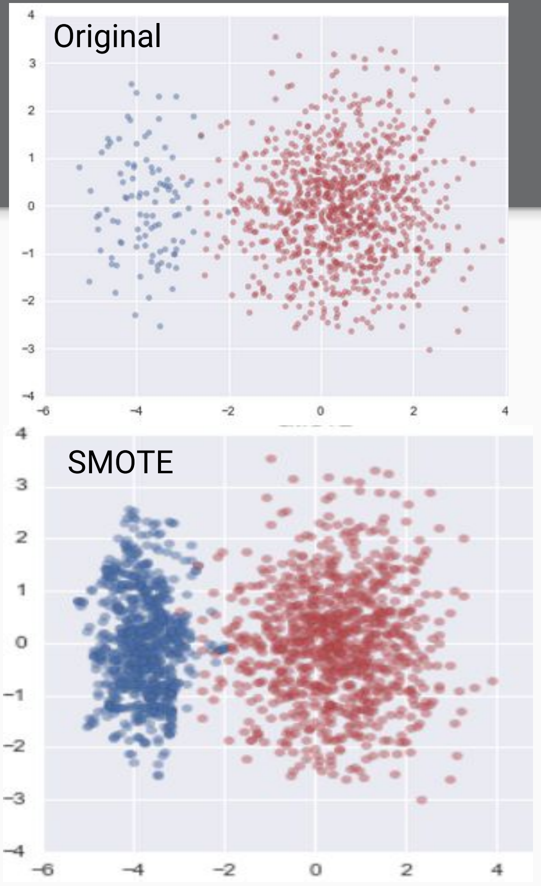
(Often better to use SMOTE)



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

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.



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

galvanize

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)
```

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...

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

Review

Practical steps:

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

Cost-sensitive learning:

- thresholding (aka “profit curves”)

Sampling:

- Oversampling
- Undersampling
- SMOTE

Best: Can you get more data?!?

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