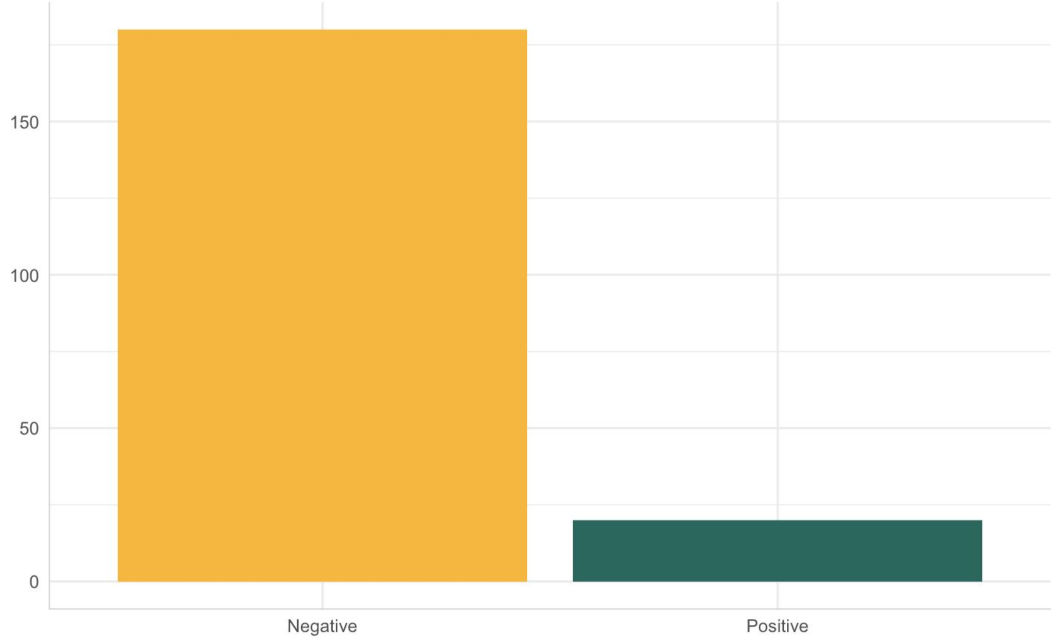
Machine Learning with Class Imbalance

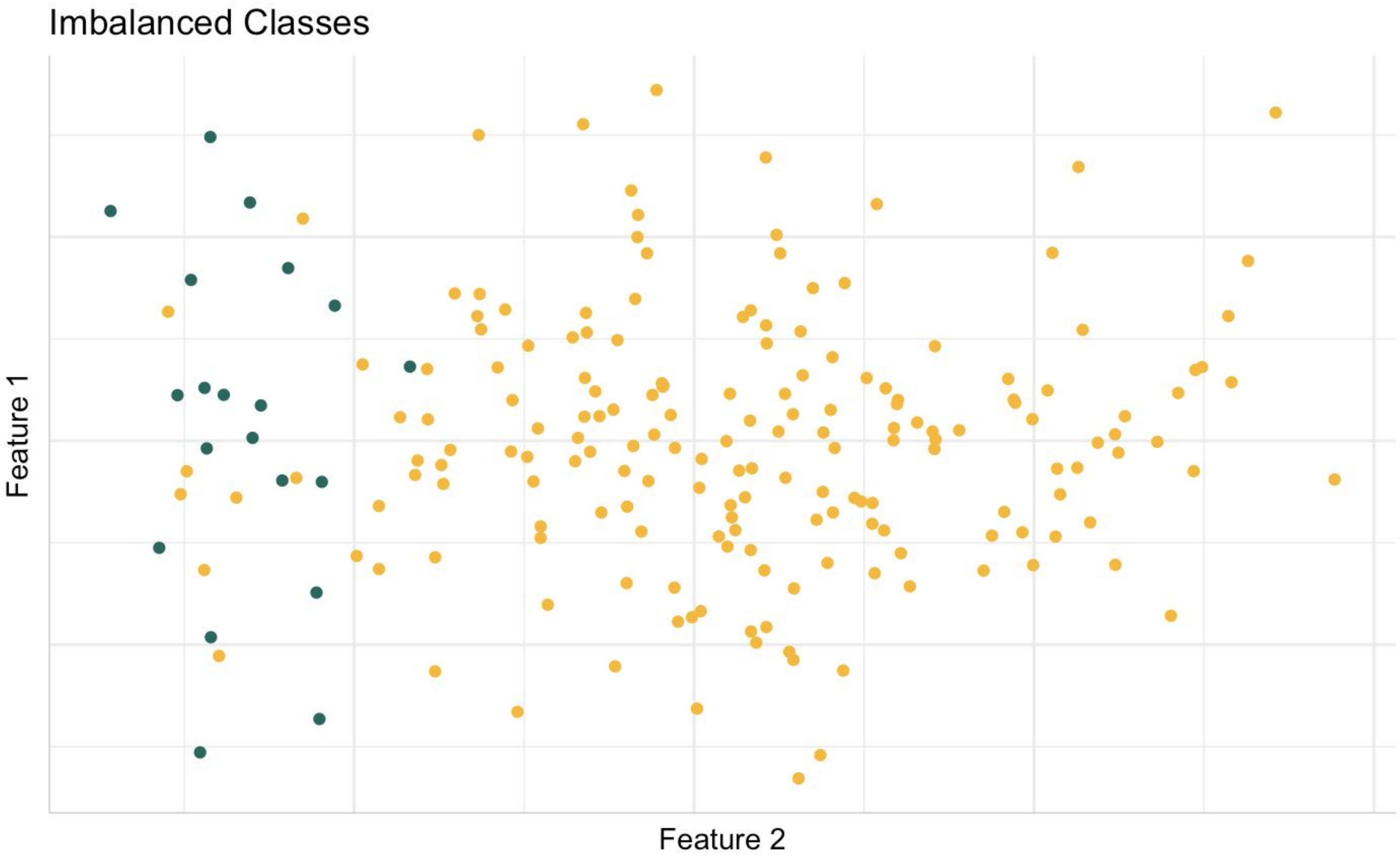
Hi, I'm Sushmita!

- Data Scientist @ Northwestern Neighborhood & Network Initiative (**@N3Initiative**)
- MA Computational Social Science, University of Chicago (10/10 would recommend)
- BA/MA Economics, Indian Institute of Technology Madras
- R-Ladies Chicago
- **@SushGopalan** on Twitter
- @sushmitavgopalan16 on Github

Class Imbalance - the class of interest is much rarer than the other class(es)

- Fraud detection
- Medical diagnoses
- Risk Prediction





Data from https://github.com/scikit-learn-contrib/imbalanced-learn

Why is this a problem?

The cost of misclassifying a minority example can be higher.

- Failure to identify a fraudulent transaction
- Failure to identify an individual as high risk for post-surgical complications

Most model evaluation metrics assume balanced classes.

- Accuracy of 98% ?

- Dig deeper
  - Confusion matrix
  - Precision
  - Recall

Always look at the confusion matrix.

	Predicted : Positive	Predicted : Negative
Actual : Positive	True Positive	False Negative
Actual : Negative	False Positive	True Negative

What proportion of positive predictions made by the model were actually positive?

$$\text{Precision} = \frac{\text{True Positives}}{\text{Total Predicted Positives}}$$

Of all observations that were actually positive, how many did our model identify as positive?

$$\text{Recall} = \frac{\text{True Positives}}{\text{Total Actual Positives}}$$

- Blindly predict Positive for everything?
    - Accuracy = 10%
    - Precision on minority class = 10%
    - Recall on minority class = 100%

- Blindly predict Positive for everything?
  - Accuracy = 10%
  - Precision on minority class = 10%
  - Recall on minority class = 100%

- Blindly predict Negative for everything?
  - Accuracy = 90%
  - Precision on minority class = NA
  - Recall on minority class = 0%

imbalance: Preprocessing Algorithms for Imbalanced Datasets

Class imbalance usually damages the performance of classifiers. Thus, it is important to treat data before applying a class
2014) <doi:10.1109/tkde.2012.232>; (Das et al. 2015) <doi:10.1109/tkde.2014.2324567>, (Zhang et al. 2014) <doi:10.1(
2014) <doi:10.1007/s00500-014-1484-5>. It also includes an useful interface to perform oversampling.

Version:	1.0.0
Depends:	R (≥ 3.3.0)
Imports:	bnlearn, KernelKnn, ggplot2, utils, stats, mvtnorm, Rcpp, smotefamily, FNN, C50
LinkingTo:	Rcpp, RcppArmadillo
Suggests:	testthat, knitr, rmarkdown
Published:	2018-02-18
Author:	Ignacio Cordón [aut, cre], Salvador García [aut], Alberto Fernández [aut], Francisco Herrera [aut]
Maintainer:	Ignacio Cordón <nacho.cordon.castillo at gmail.com>
BugReports:	http://github.com/ncordon/imbalance/issues
License:	GPL-2 | GPL-3 | file LICENSE [expanded from: GPL (≥ 2) | file LICENSE]
URL:	http://github.com/ncordon/imbalance
NeedsCompilation:	yes
Materials:	README
CRAN checks:	imbalance results

unbalanced: Racing for Unbalanced Methods Selection

A dataset is said to be unbalanced when the class of interest (minority class) is much rarer than
Most learning systems are not prepared to cope with unbalanced data and several techniques ha
adaptively the most appropriate strategy for a given unbalanced task.

Version:	2.0
Depends:	mlr, foreach, doParallel
Imports:	FNN, RANN
Suggests:	randomForest, ROCR
Published:	2015-06-26
Author:	Andrea Dal Pozzolo, Olivier Caelen and Gianluca Bontempi
Maintainer:	Andrea Dal Pozzolo <adalpozz at ulb.ac.be>
License:	GPL (≥ 3)
URL:	http://mlg.ulb.ac.be
NeedsCompilation:	no
CRAN checks:	unbalanced results

build passing | build passing | codecov 98% | circleci passing | docs passing | python 2.7 \| 3.6 \| 3.7 | pypi package 0.4.3 | chat on gitter

imbalanced-learn

imbalanced-learn is a python package offering a number of re-sampling techniques commonly used in datasets showing strong between-class imbalance. It is compatible with scikit-learn and is part of scikit-learn-contrib projects.

Documentation

Installation documentation, API documentation, and examples can be found on the documentation.

Installation

Dependencies

imbalanced-learn is tested to work under Python 2.7 and Python 3.6, and 3.7. The dependency requirements are based on the last scikit-learn release:

- scipy(>=0.13.3)
- numpy(>=1.8.2)
- scikit-learn(>=0.20)
- keras 2 (optional)
- tensorflow (optional)

What can you do?

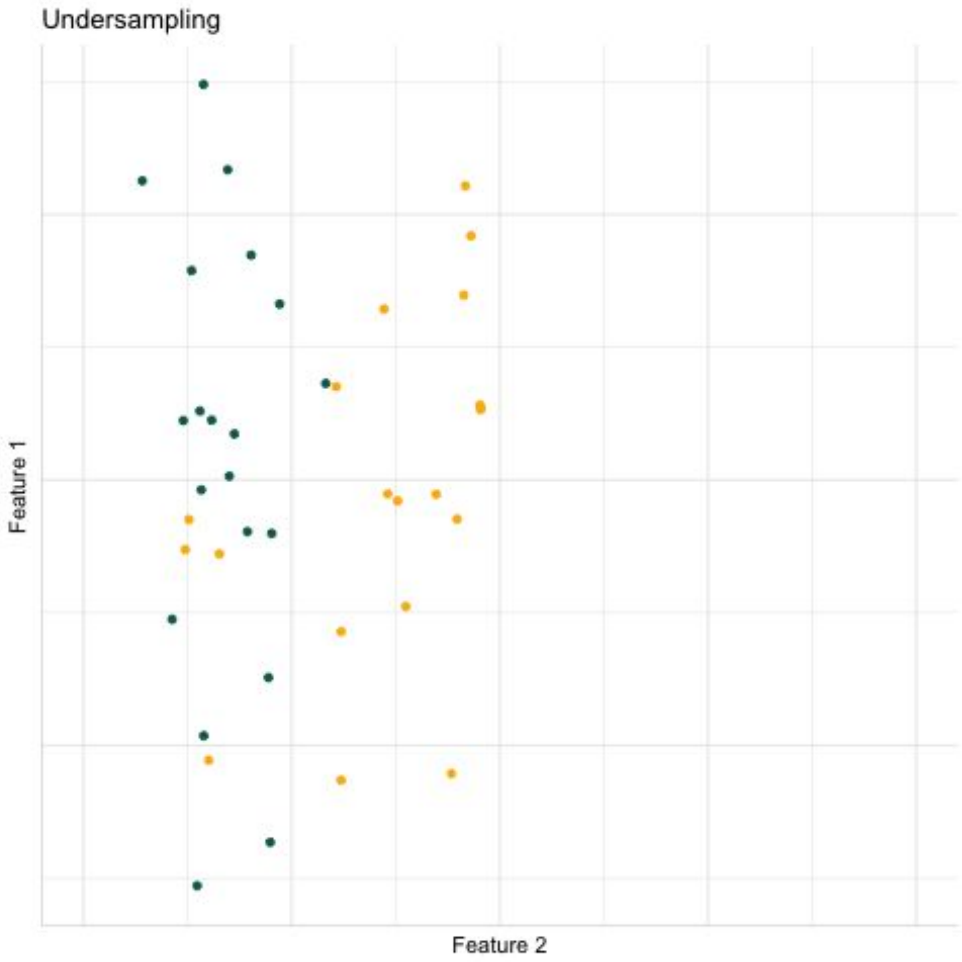
- ‘Data Mining’ Solutions

- Machine Learning Solutions
    - Cost sensitive learning
    - Ensemble methods

Random Undersampling

```
minority <- data %>% filter(y == 1)
majority <- data %>% filter(y == 0)

undersampled <- sample_n(majority, nrow(minority)) %>%
                bind_rows(minority)
```

```
library(unbalanced)
undersampled <- ubUnder(X = features,
        Y = labels,
        perc = round(nrow(minority)*100 / nrow(majority)),
        method = 'percUnder')
```

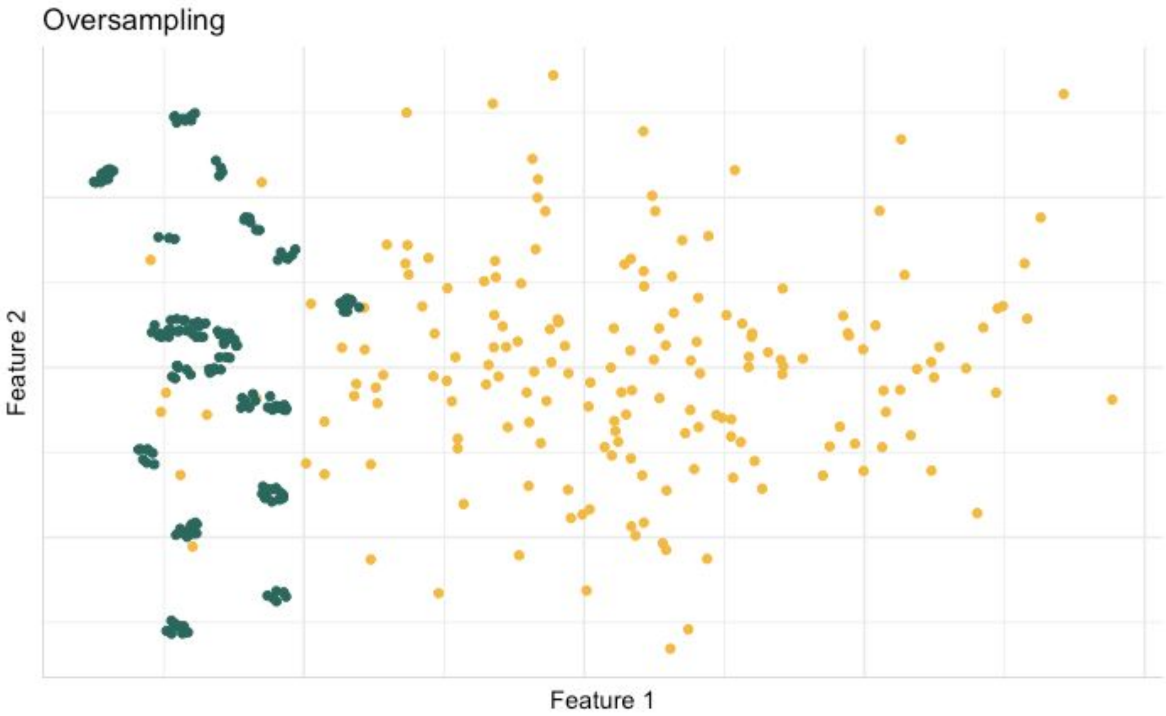


- Throwing away good data
- Decision boundary changes with chosen subset
- Cross-validate

Random Oversampling

```
minority <- data %>% filter(y == 1)
majority <- data %>% filter(y == 0)

oversampled <- sample_n(minotiy, nrow(majority), replace = TRUE) %>%
                bind_rows(majority)
```

```
library(unbalanced)
oversampled <- ubOver(X = features,
                      Y = label)
```

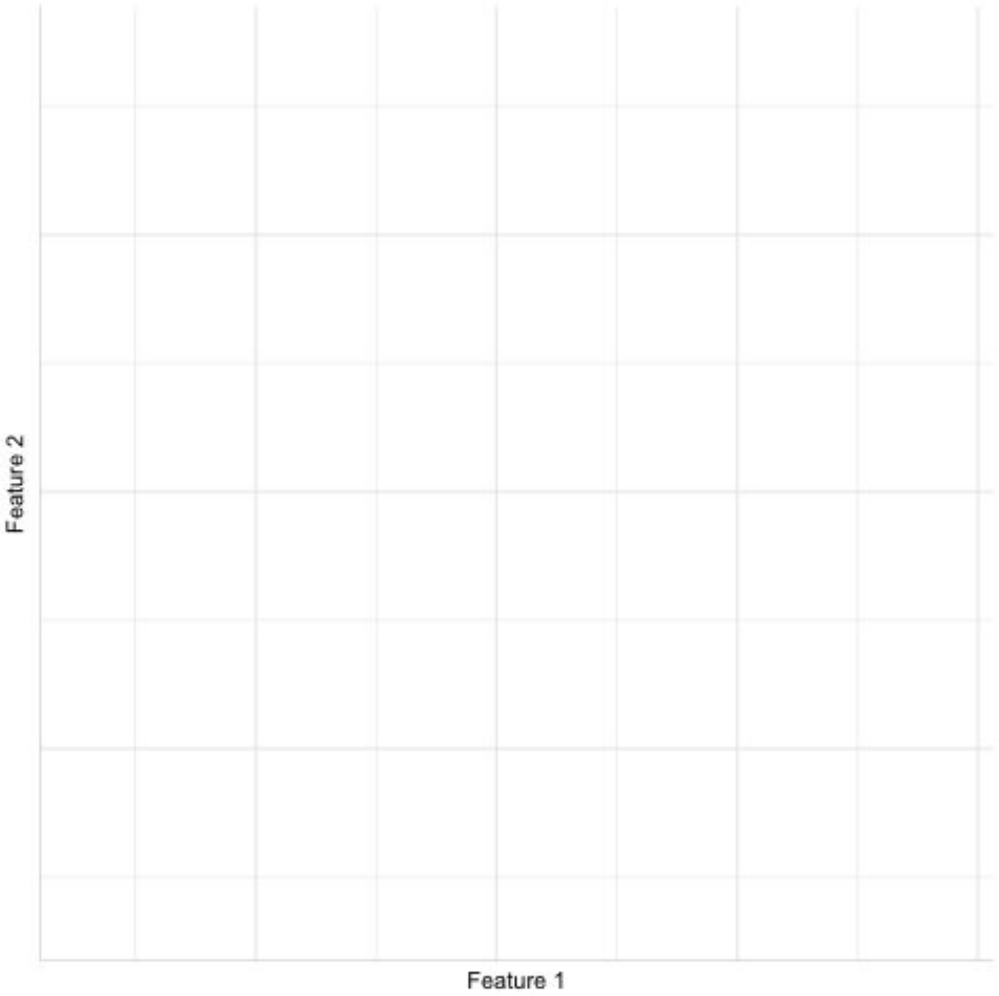


- Equivalent to re-weighting (with a little randomness)
- More data ≠ more information
- Undue emphasis on outliers
- Risk of overfitting

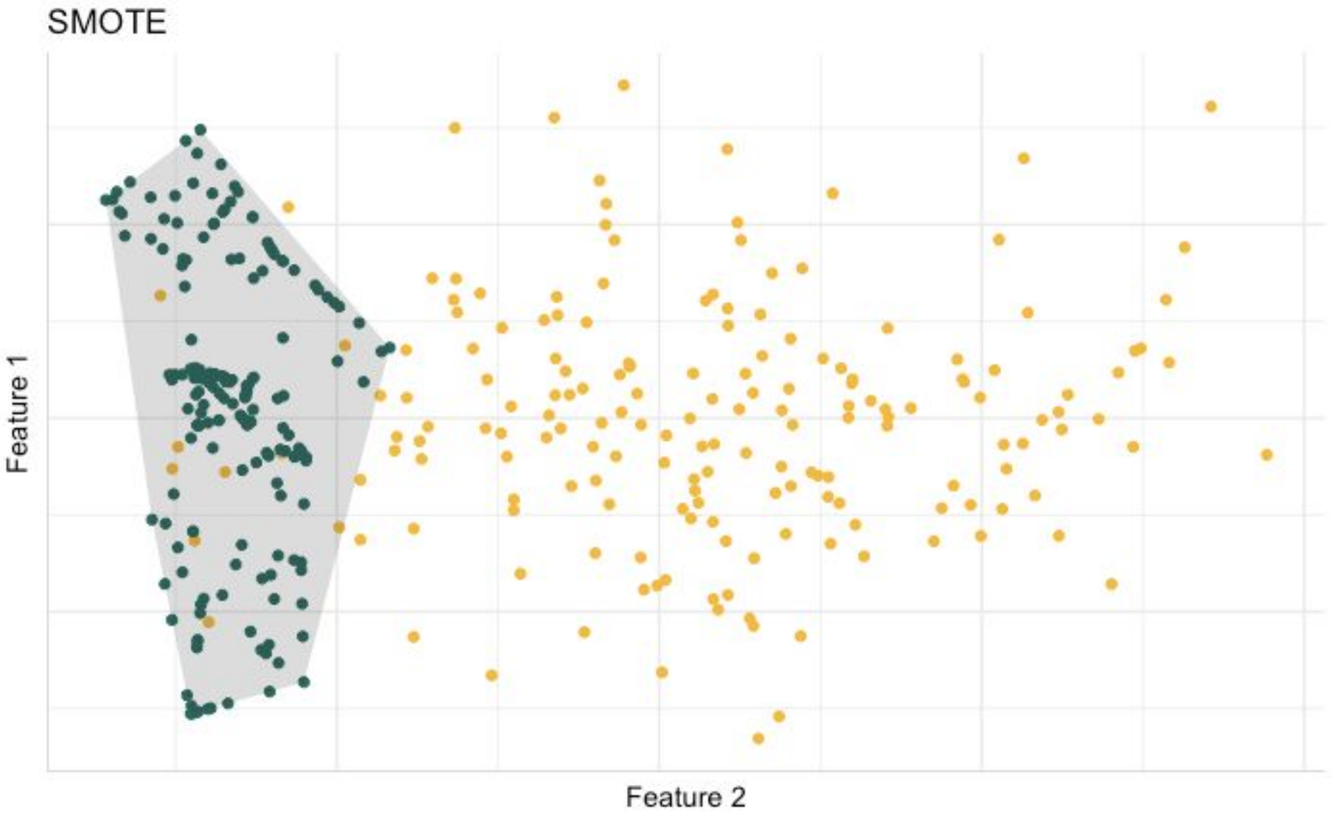
Synthetic Minority Oversampling Technique (SMOTE)

```
library(unbalanced)

perc_over <- nrow(majority)*100/nrow(minority)

smoted_data <- ubSMOTE(X = features,
                       Y = labels,
                       perc.over = perc_over)
```

Chawla, Nitesh V, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," Journal of Artificial Intelligence Research, 2002, 16, 321–357.



- New data points through interpolation
- Less overfitting
- New points lie within same convex space

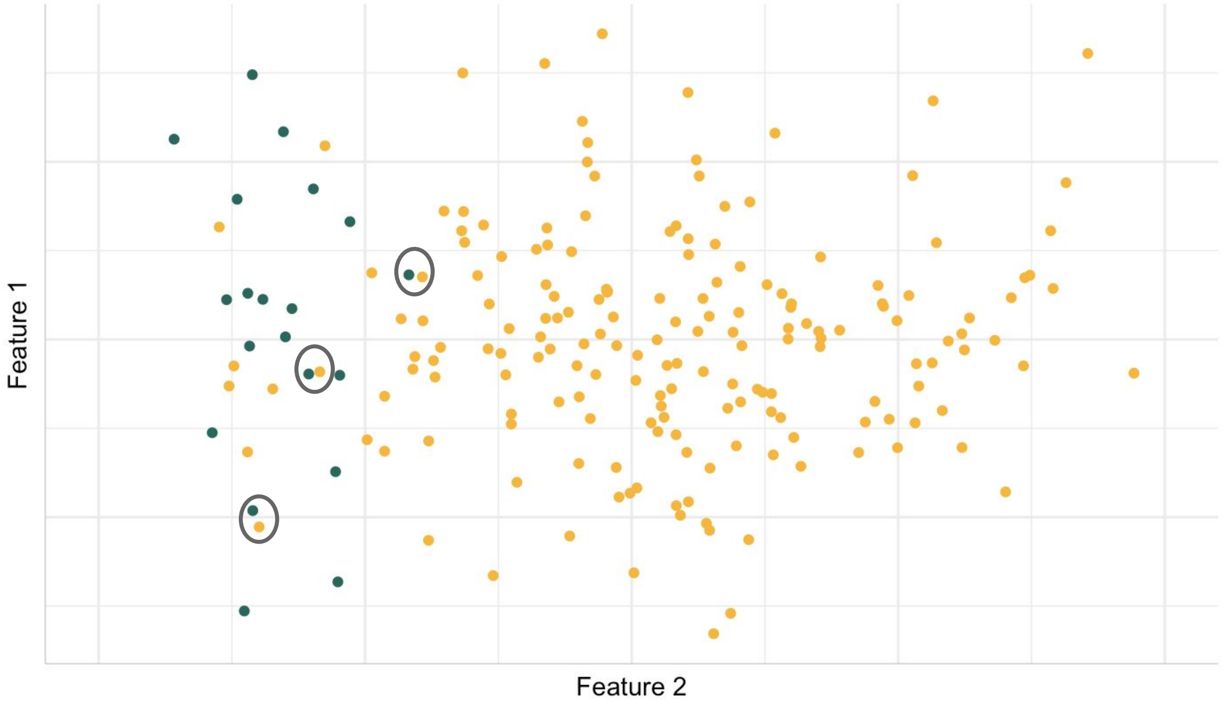

SMOTE
Feature 1
Feature 2

Removing TOMEK Links

- Disambiguate class boundaries

- A pair of observations forms a TOMEK link if
  - They are each other's **nearest neighbour** and
  - They have **different** class labels

Tomek, Ivan, "Two modifications of CNN," IEEE Trans. Systems, Man and Cybernetics, 1976, 6, 769–772.

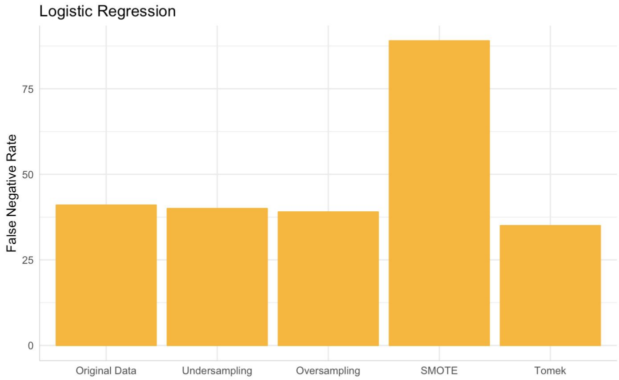


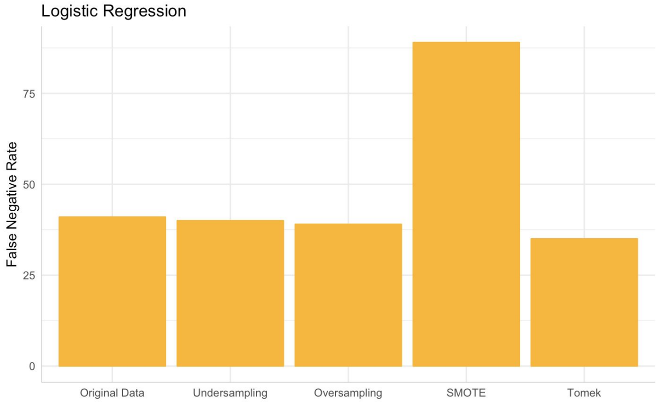
- Removes noisy borderline examples
- Works best in conjunction with some oversampling / adaptive learning algorithms

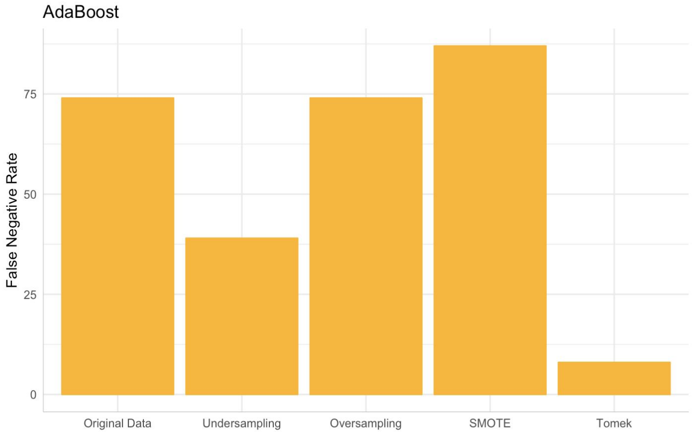
Predicting Infant Mortality: Minimizing False Negatives

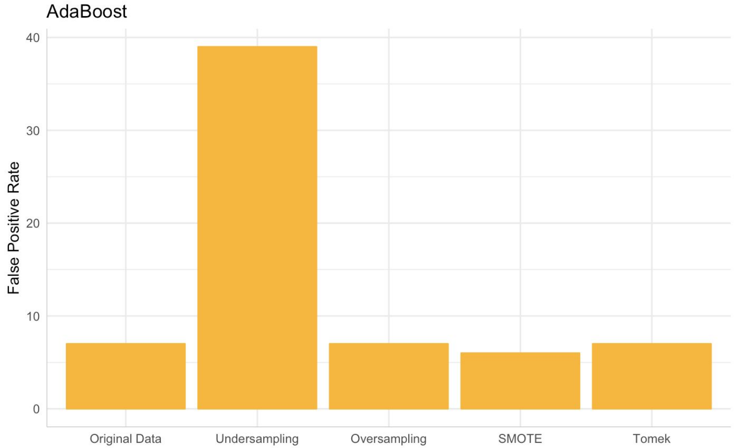
- **32 out of 1,000** babies born in India die within a year
- Can we identify pregnant mothers at risk for infant death?
- Some predictive variables -
  - Mother's age
  - Time since previous birth
  - Anaemia

- **92.7 %** accuracy from vanilla logistic regression
- However, I was misclassifying **3 out of 4** high-risk pregnancies as safe!
- Focus on minimizing False Negatives?
- What are the trade-offs?
- Class Imbalance Ratio - **93:7**


Logistic Regression
False Negative Rate
0
25
50
75
Original Data
Undersampling
Oversampling
SMOTE
Tomek


Logistic Regression
False Negative Rate
0
25
50
75
Original Data
Undersampling
Oversampling
SMOTE
Tomek


AdaBoost
False Negative Rate
0
25
50
75
Original Data
Undersampling
Oversampling
SMOTE
Tomek


AdaBoost
False Positive Rate
0
10
20
30
40
Original Data
Undersampling
Oversampling
SMOTE
Tomek

Why did removing Tomek Links + AdaBoost work best for THIS data, given THESE objectives?

- Discarded only 0.2% of the data
- Robust to cross-validation
- Sick babies that survive can look very similar to sick babies that do not

How do you choose a strategy?

- Know the structure of your data
- Evaluate your trade-offs
- Trial-and-error

Thank you!