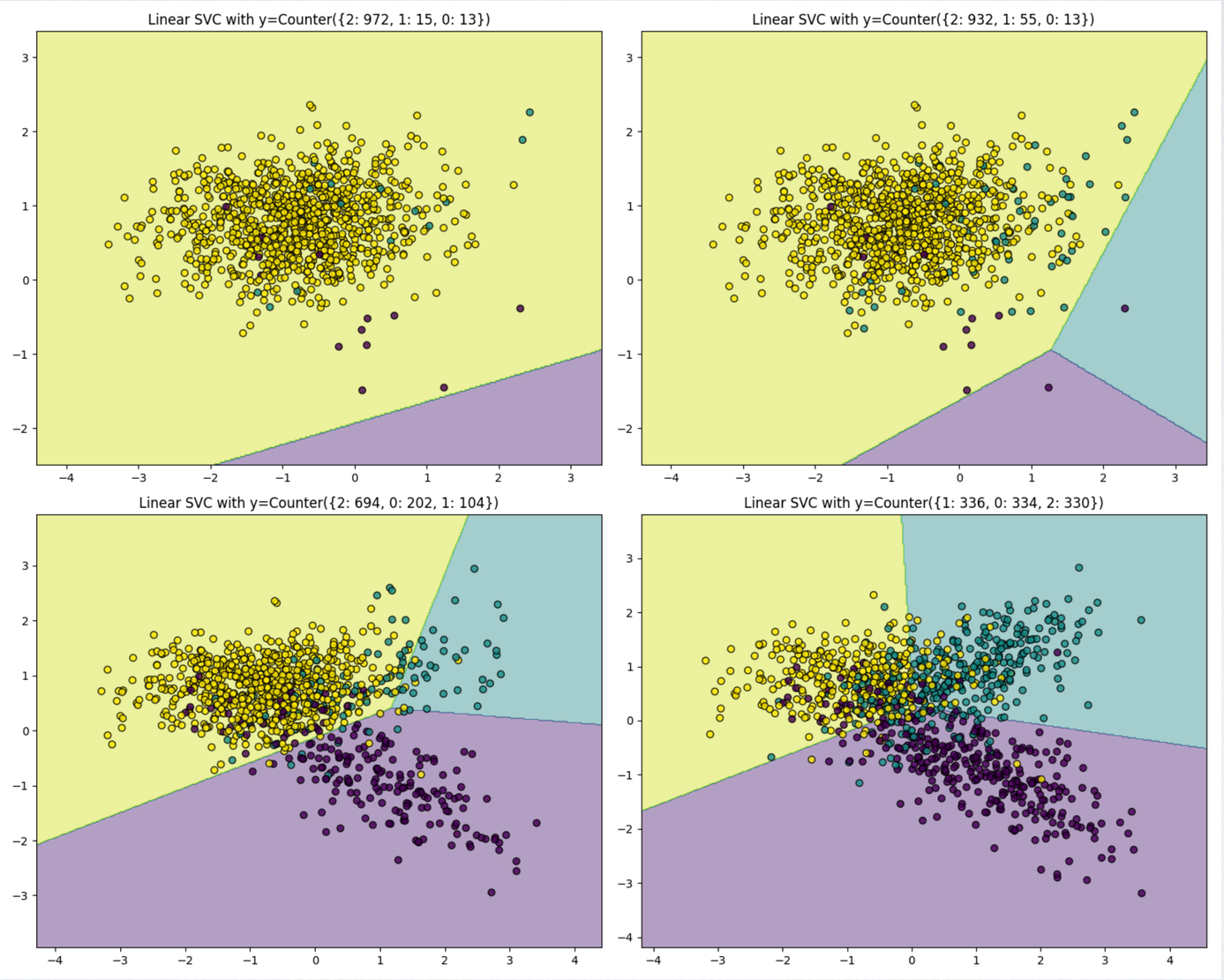
Managing Class Imbalance With Imbalanced-Learn

Chad Scherrer

The Problem


Linear SVC with y=Counter({2: 972, 1: 15, 0: 13})
Linear SVC with y=Counter({2: 932, 1: 55, 0: 13})
Linear SVC with y=Counter({2: 694, 0: 202, 1: 104})
Linear SVC with y=Counter({1: 336, 0: 334, 2: 330})

Lots of Approaches

- General sklearn approaches
- Oversampling
- Undersampling
- Combination
- Ensembles
- Check out http://contrib.scikit-learn.org/imbalanced-learn

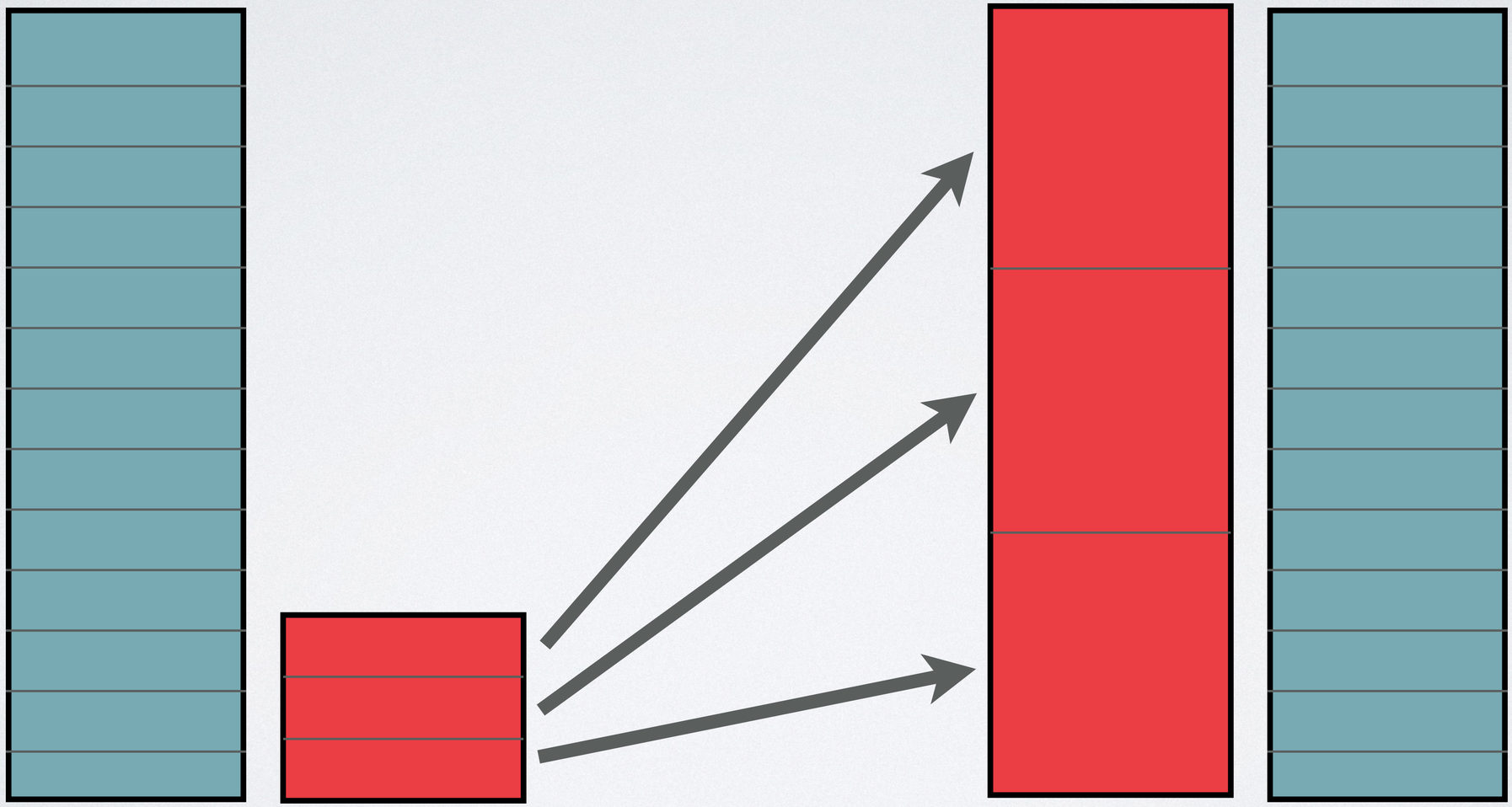
Weighting

- Many models allow weighted observations
- Adjust these so total weights are equal across classes
- Easy to do, when it’s available
- No need to sacrifice data
- Also possible with SGD optimization (reweight probabilities)

Stratified Sampling

- Train-test split, “stratify” option
- ShuffleSplit -> StratifiedShuffleSplit
- KFold -> StratifiedKFold -> RepeatedStratifiedKFold

Oversampling

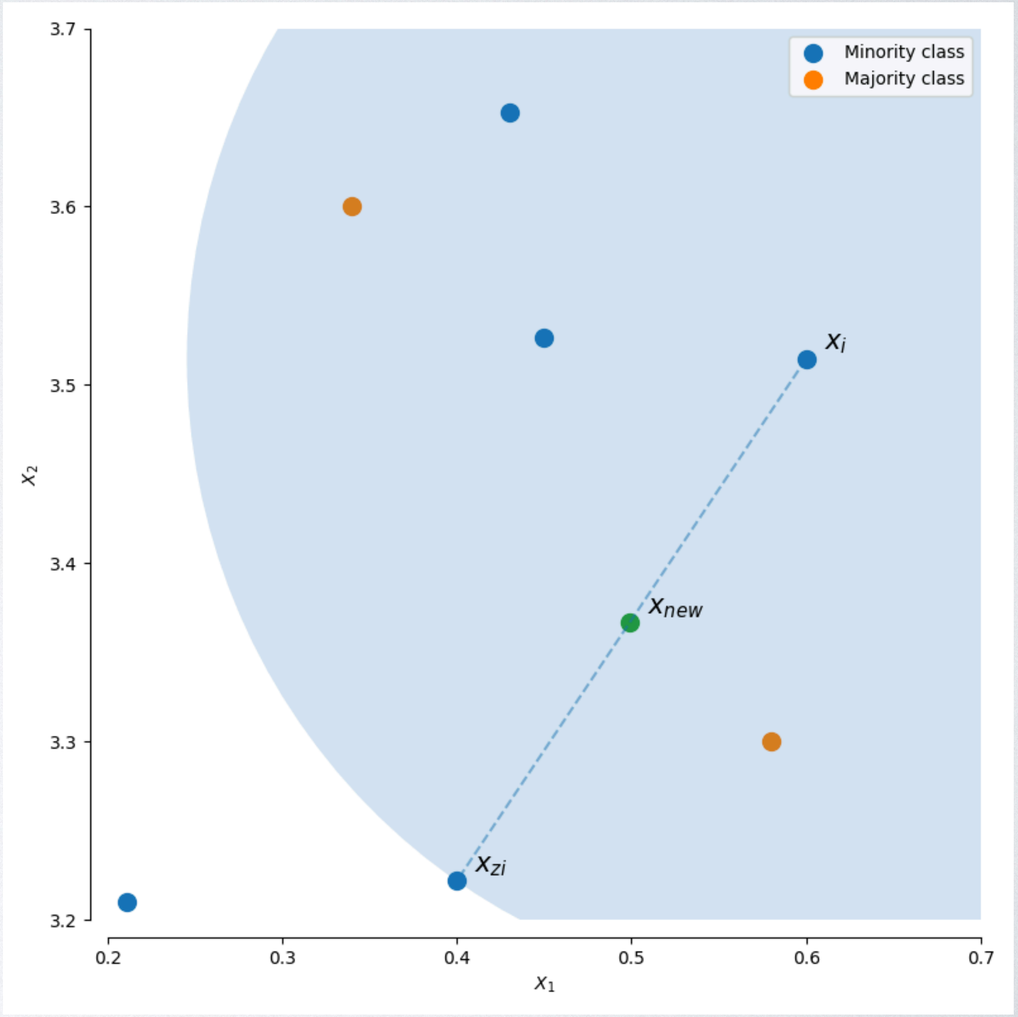


Random Oversampling

- Simplest oversampling approach
- Resample with replacement from minority class
- No concerns about geometry of feature space
- Good for categorical data

Synthetic Oversampling

- Start with a point in the minority class
- Choose one of K nearest neighbors (how?)
- Add a new point between them
- Two main approaches:
    - SMOTE
    - ADASYN



http://contrib.scikit-learn.org/imbalanced-learn/stable/over_sampling.html#a-practical-guide

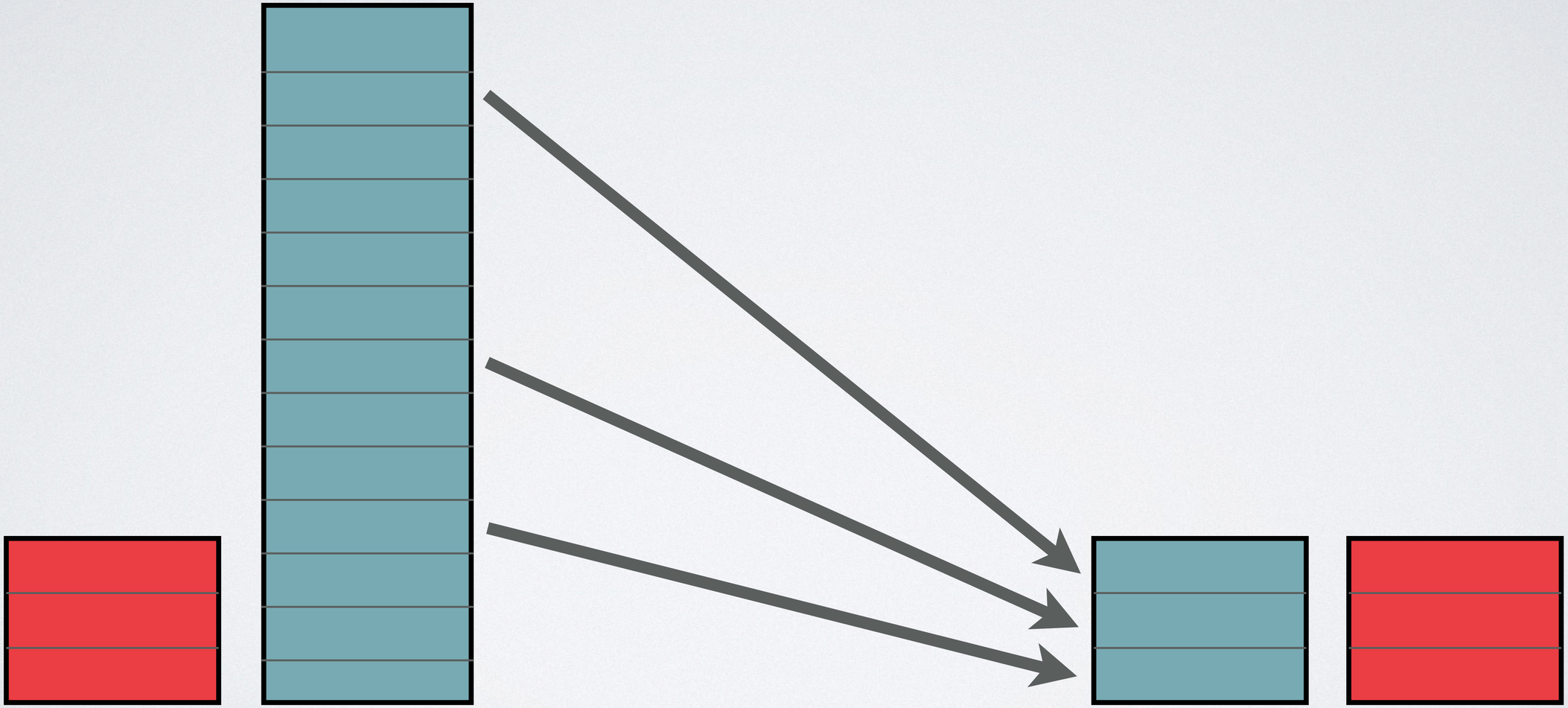
SMOTE:
Synthetic Minority Oversampling Technique

- **Regular:** Plain ol' SMOTE. Connect minority class points to any neighbor (even other classes)
- **Borderline:** Classify points as *outlier*, *safe*, or *in-danger*
    - **1:** Connect minority *in-danger* points only to minority points
    - **2:** Connect minority *in-danger* points to whatever is nearby
- **SVM:** Use minority support vectors to generate new points

ADASYN:
ADAptive SYNthetic Sampling

- For each minority point,
- Look at classes in neighborhood
- Generate new samples proportional to competing classes
- Motivated by KNN, but helps other classifiers as well

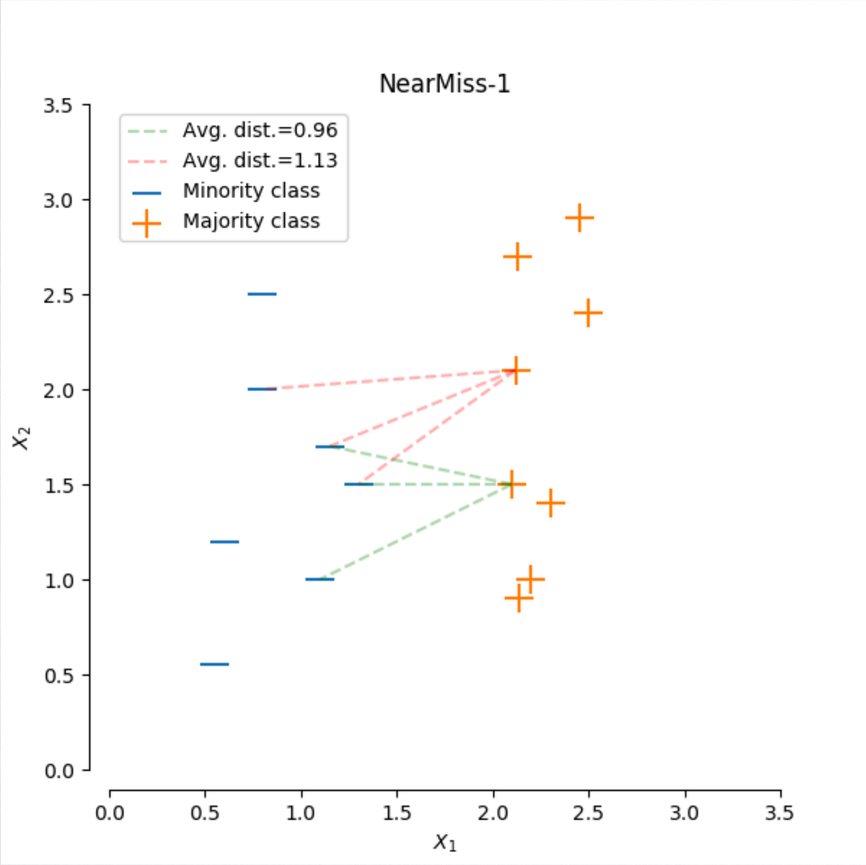
Undersampling

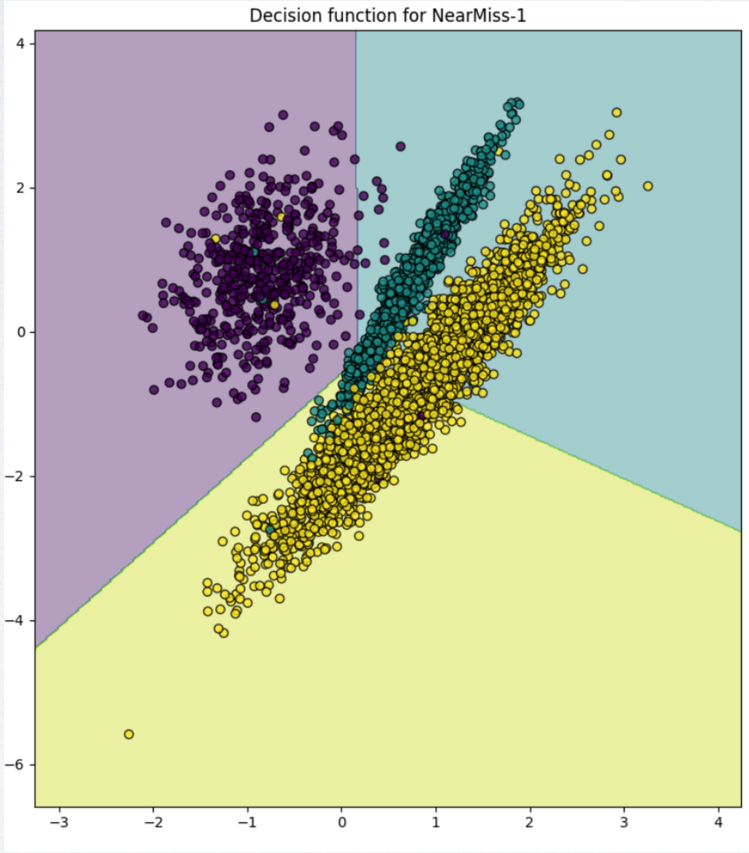


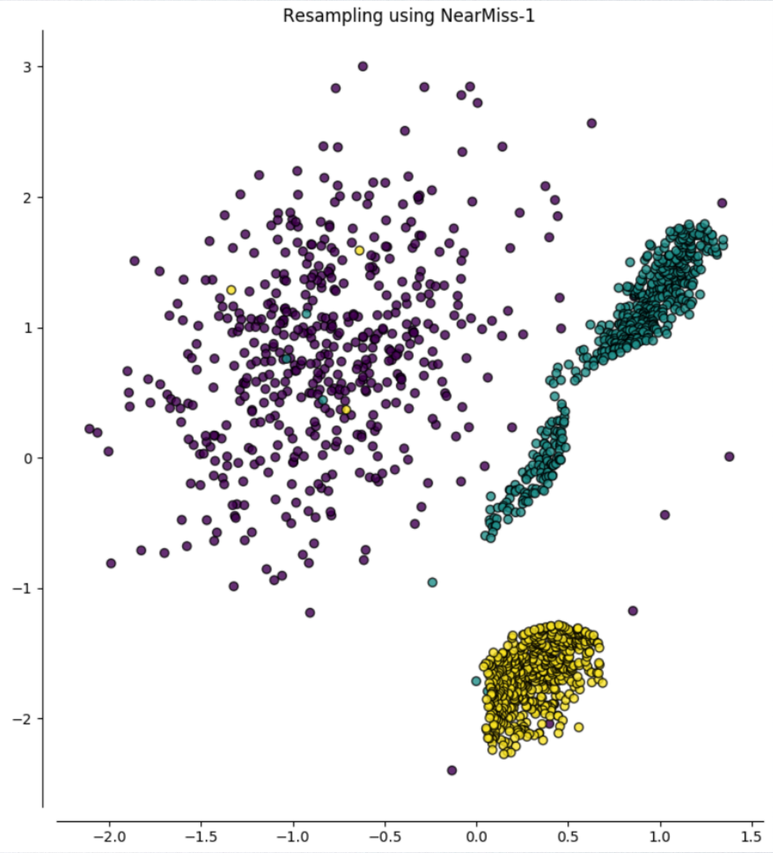
Prototype Generation: Cluster Centroids

- Generate (fewer) new representative points
- Replace original points with these

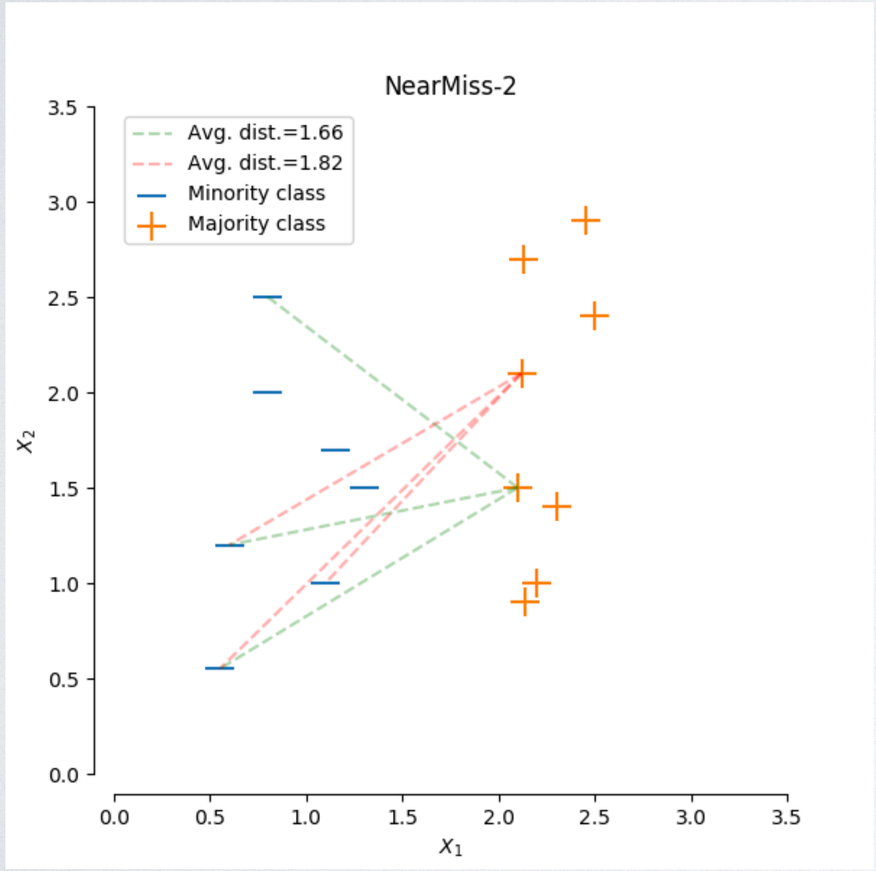
NearMiss-1: Which Points To Keep? Those Closest to Nearby Minority Points

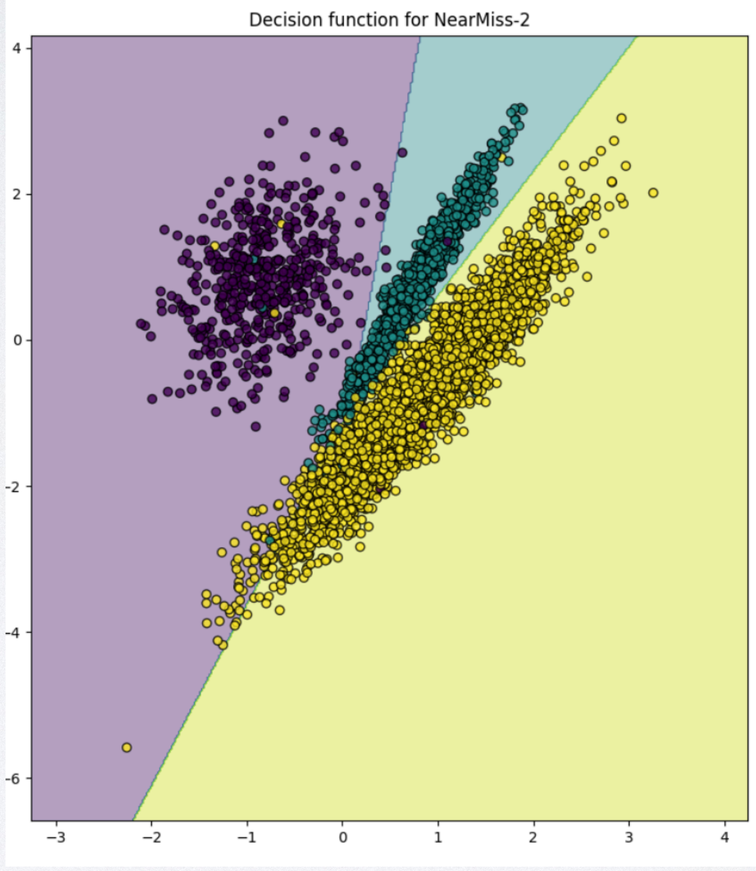


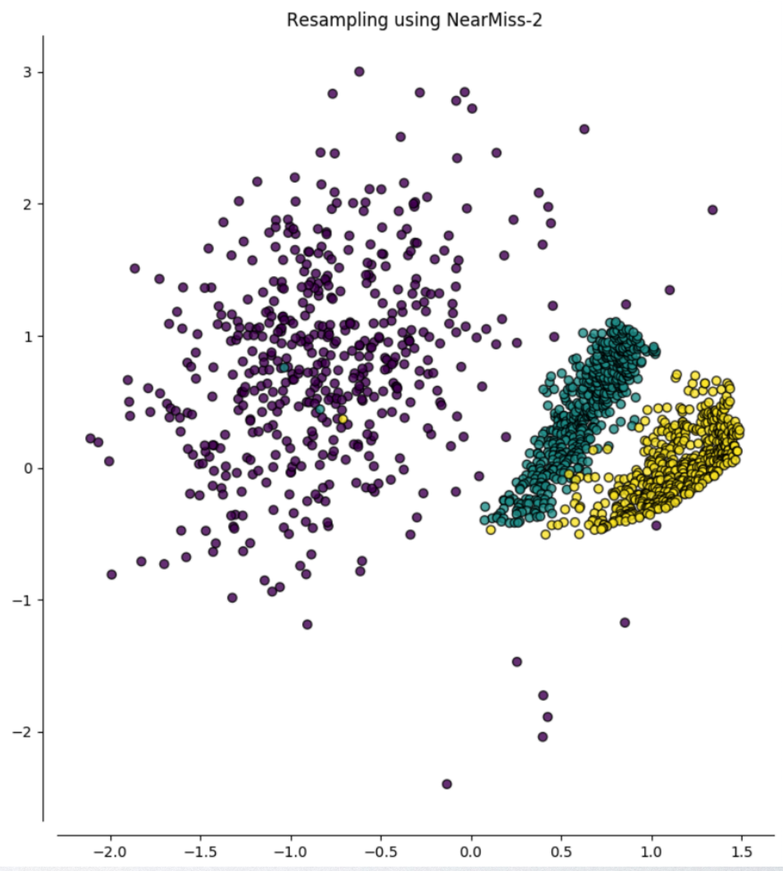




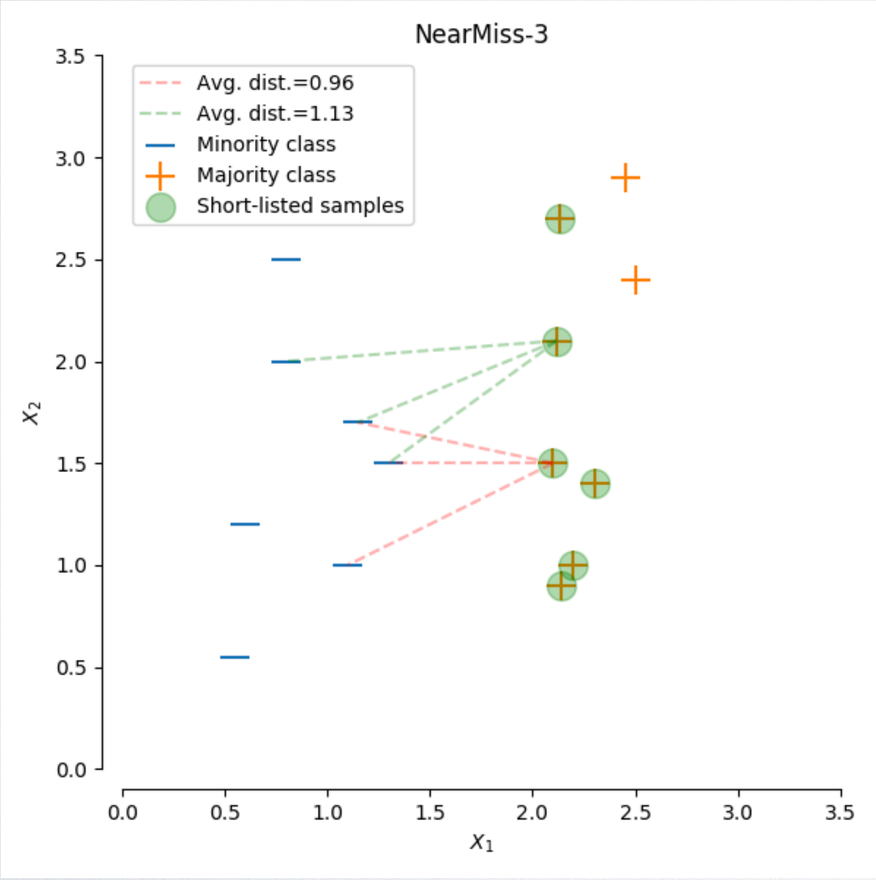
NearMiss-2: Which Points To Keep? Those Closest to Distant Minority Points

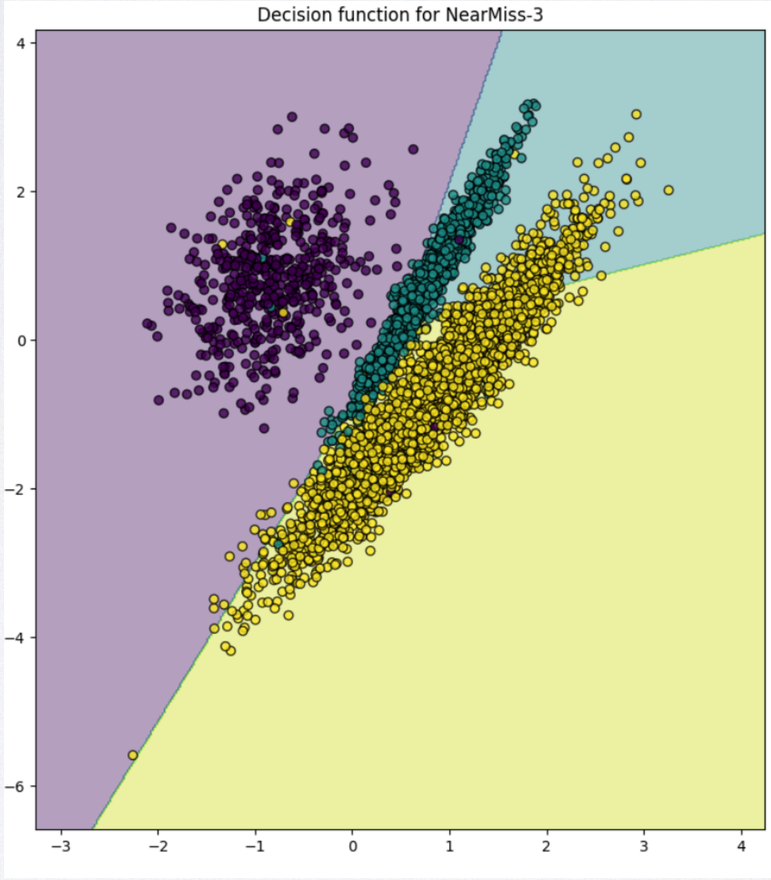


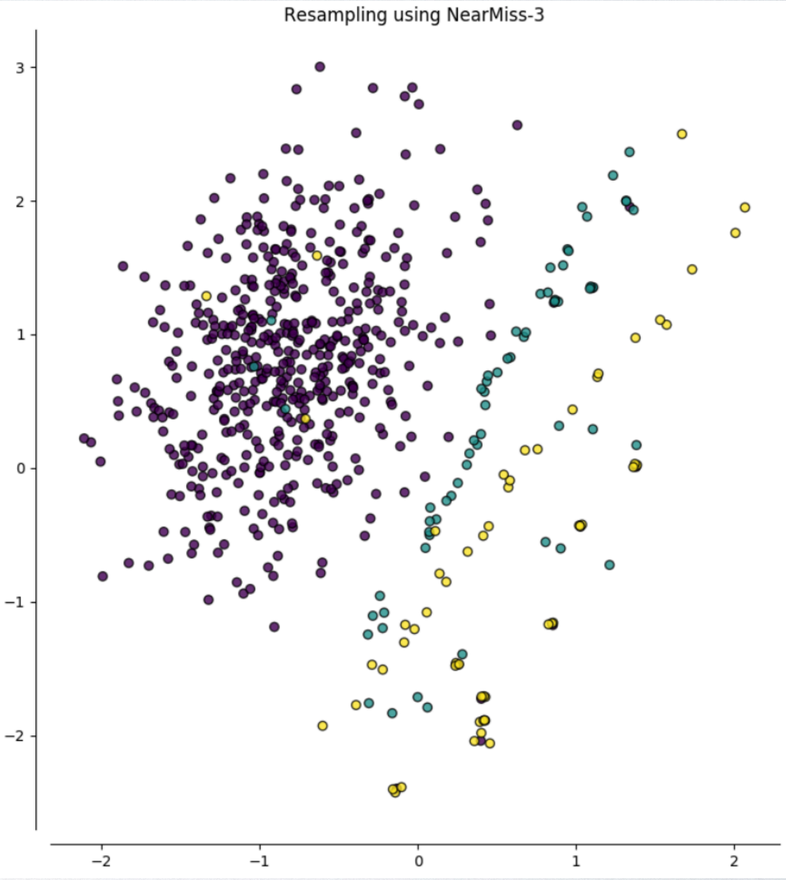




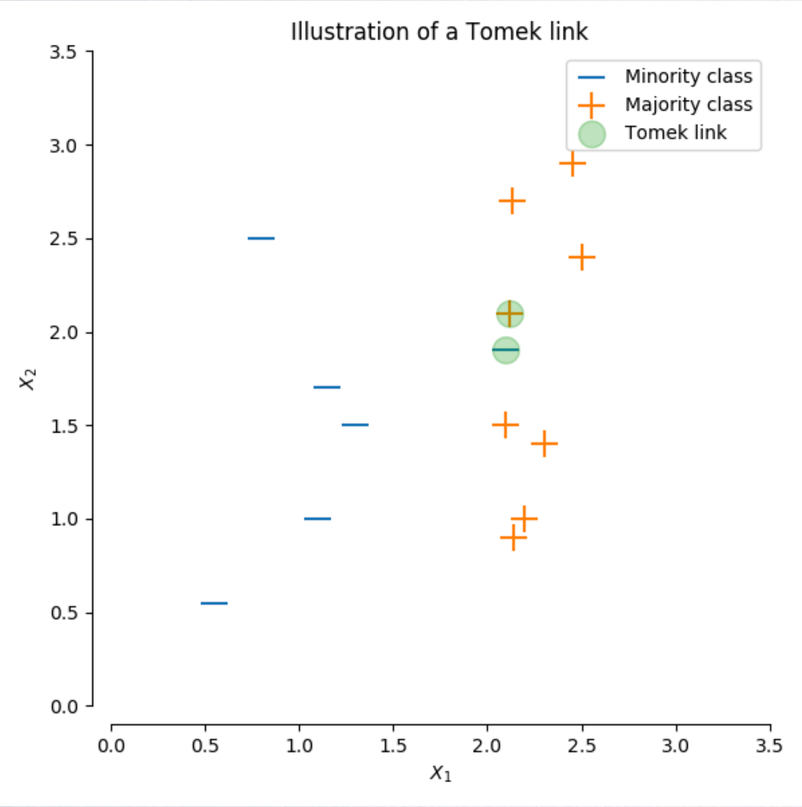
NearMiss-3: Those Closest to Majority Neighbors of Minority Points

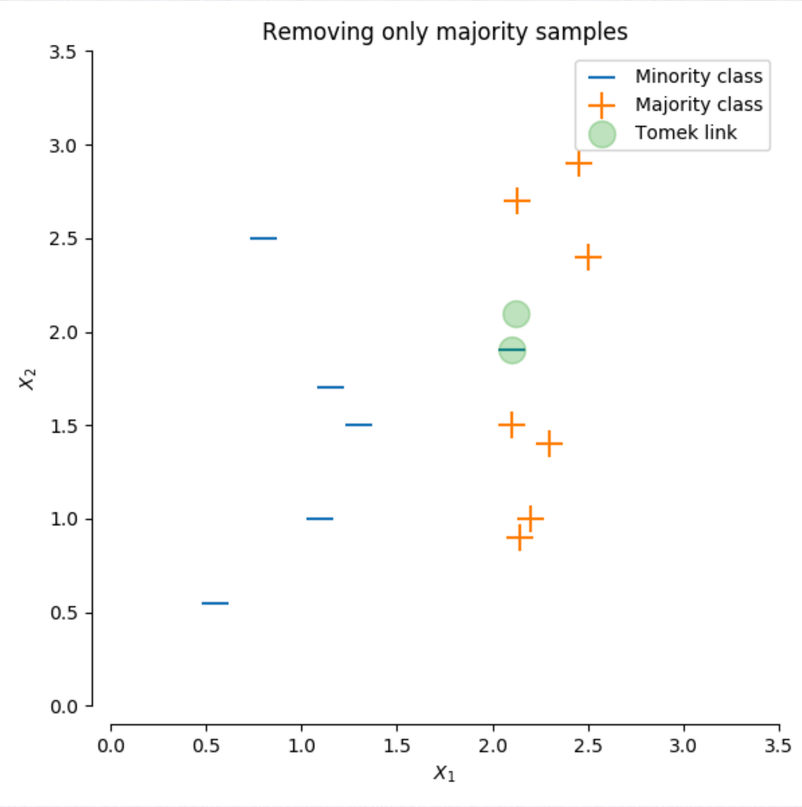


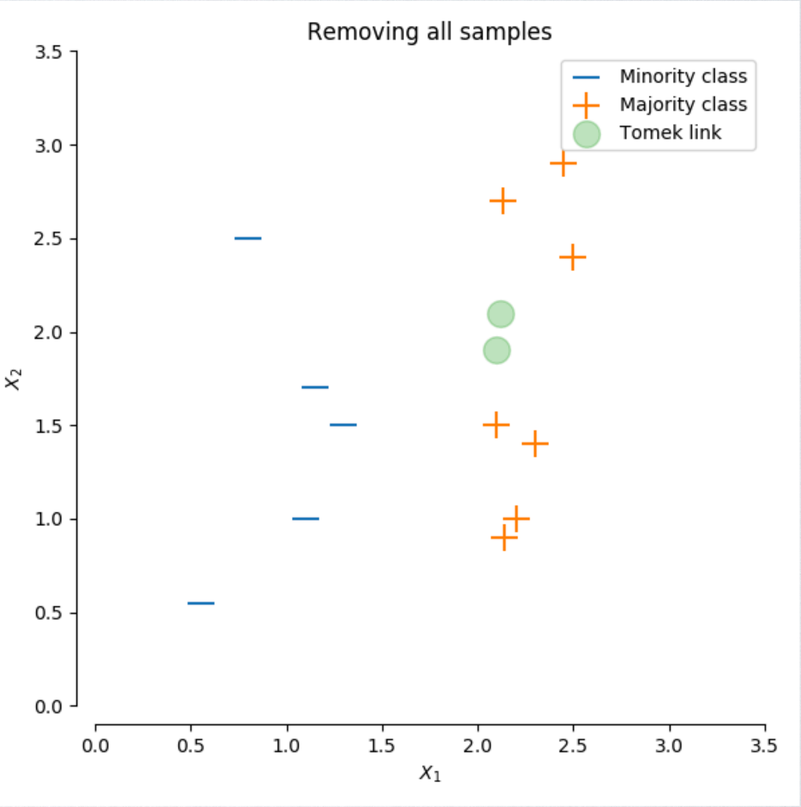




Tomek Links: Mixed Mutual Nearest Neighbors

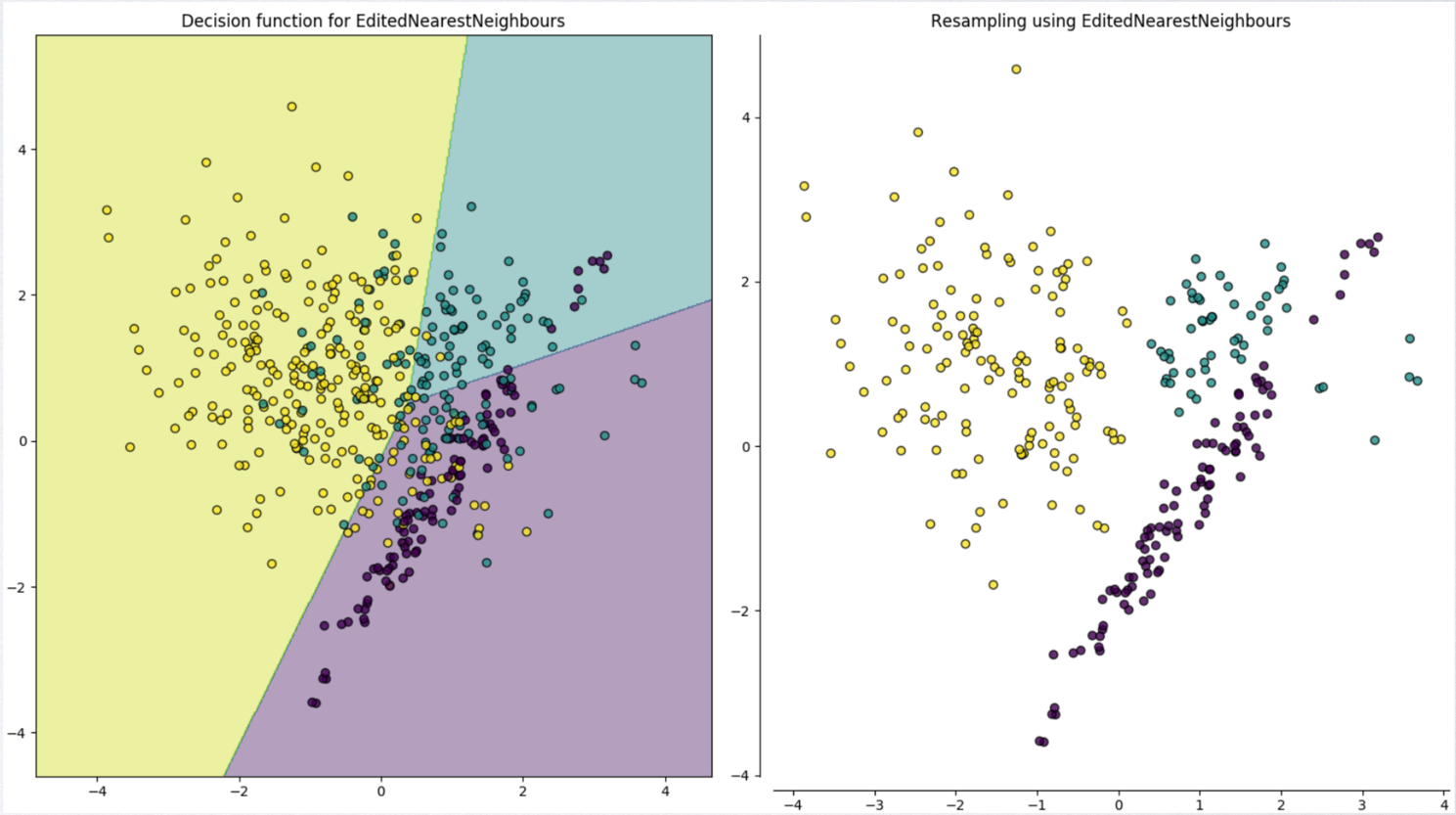






Edited Nearest Neighbors

- Remove points that don't agree with neighbors

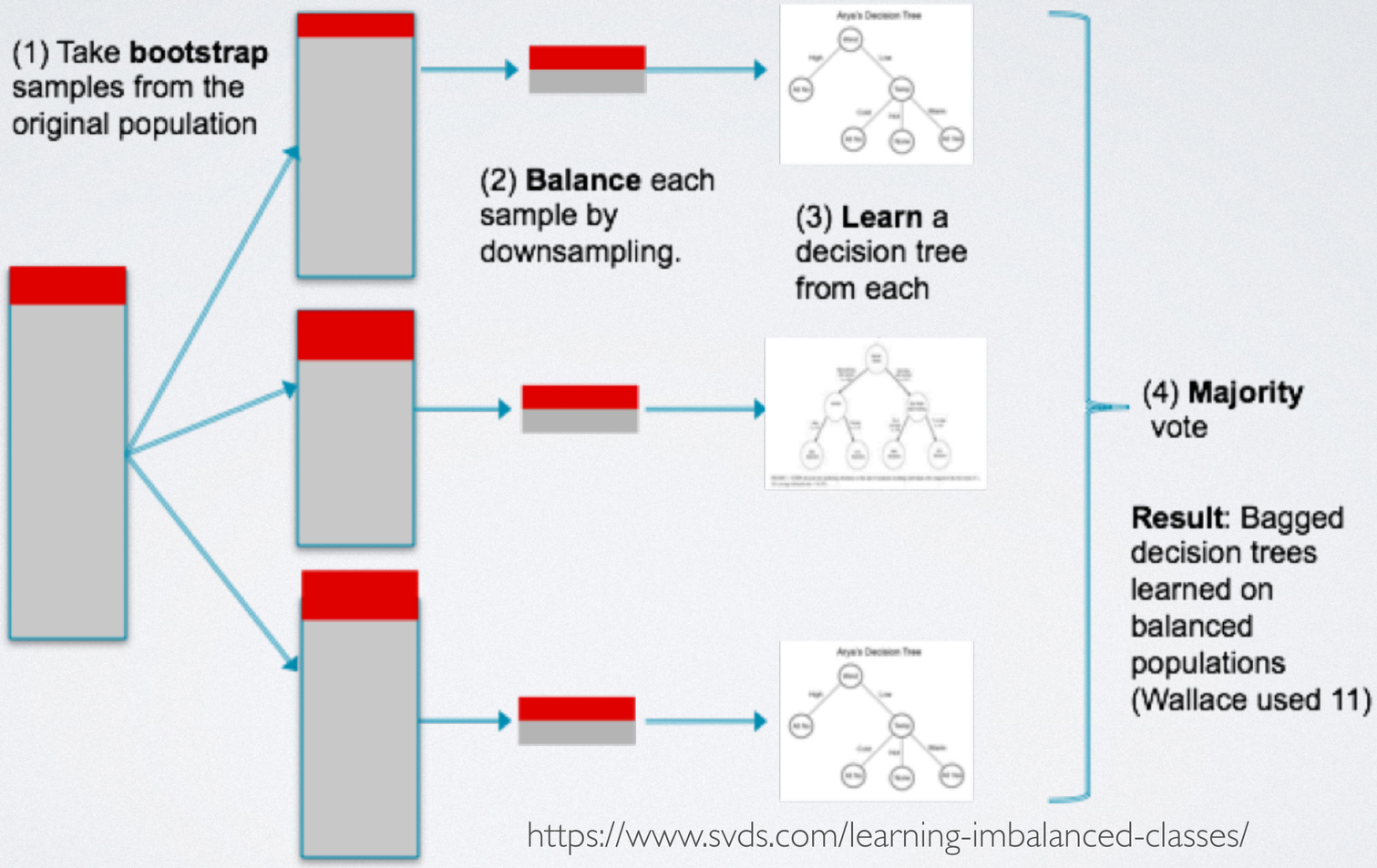


See also: RepeatedEditedNearestNeighbors, AllKNN

Combination Over/Under

- SMOTE + Tomek’s link
- SMOTE + Edited Nearest Neighbors

Blagging (Balanced Bagging)



https://www.svds.com/learning-imbalanced-classes/

Final Words

- All of this happens **after** the test set has been split.
- Use sensible metrics
    - AUC
    - F1
    - Cohen’s Kappa
    - **Not** accuracy - too easy to fool in this case