Imbalanced Data

1. What is imbalanced data
2. Preprocessing methods
3. Examples of classification algorithms

Mariusz Wiśniewski

What is imbalanced data

Imbalanced datasets are those in which examples from some classes occur much more often than from other classes.

- fraud detection
- medical diagnosis
- product categorization
- network intrusion detection

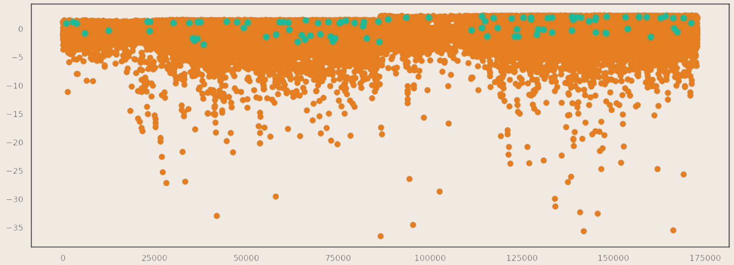
Preprocessing methods

- Oversampling methods
  - Random
  - SMOTE
  - Borderline1-SMOTE
  - Borderline2-SMOTE
- Undersampling methods
  - Random
  - Near Miss1-3
  - CNN
  - ENN
  - RENN
  - Tomek Link Removal

Introduction

- Markings:
  - L (large) - majority class
  - S (small) - minority class
  - r (ratio) = |S|/|L|
  - T - training set

> *Credit Fraudulent Detection*
> *|L| = 284315*
> *|S| = 492*
> *r = 0.17%*

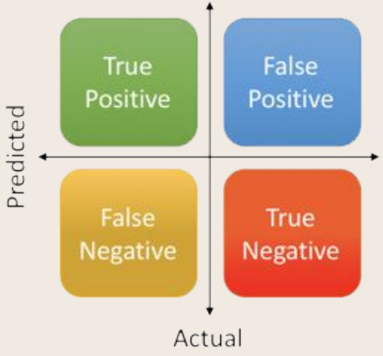


$$\textbf{Precision} = \frac{\text{True Positive}}{\text{Actual Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive + False Positive}}$$

$$\textbf{Recall} = \frac{\text{True Positive}}{\text{Predicted Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive + False Negative}}$$

$$\textbf{Accuracy} = \frac{\text{True Positive + True Negative}}{\text{Total}}$$

$$\textbf{F1 Score} = 2 * \frac{\text{Precision * Recall}}{\text{Precision + Recall}}$$



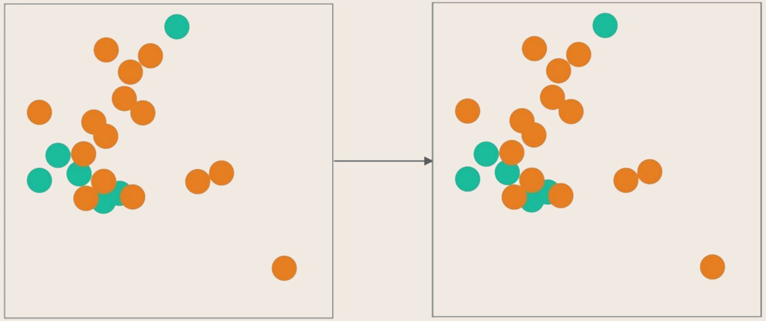
https://www.kaggle.com/mlg-ulb/creditcardfraud/home
https://github.com/Nexer8/Imbalanced_Data

Oversampling methods

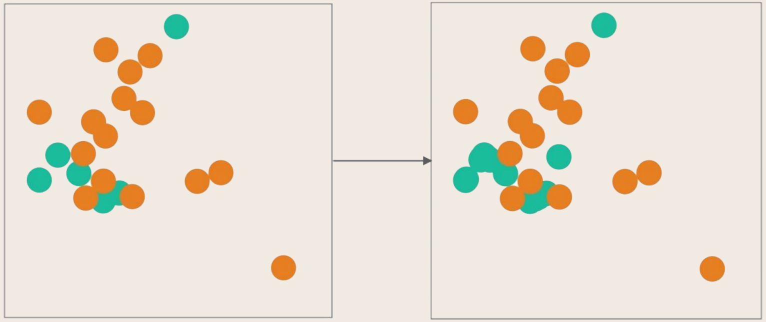
- *ROS - Random Oversampling*
  - duplication of randomly selected points from the minority class
- *SMOTE* (*Synthetic Minority Oversampling Technique*)
  1. Calculate the *k* nearest neighbors from *S*.
  2. Randomly select $r \leq k$ neighbors (with swap).
  3. Pick a random point along the lines connecting *p* and each of the *r* neighbors.
  4. Add these designated points to the *S* class dataset.

\> *Random Oversampling (|L| = |S| = 213245)*
  - *Precision = 0.08*
  - *Recall = 0.87*
  - *f1 = 0.14*

\> *SMOTE (|L| = |S| = 213245)*
  - *Precision = 0.1*
  - *Recall = 0.86*
  - *f1 = 0.18*



Random oversampling



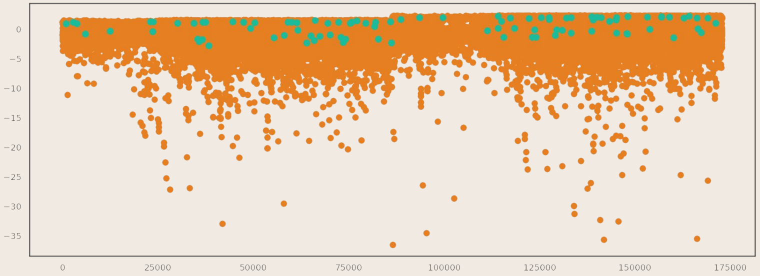
Synthetic Minority Oversampling Technique

Oversampling methods

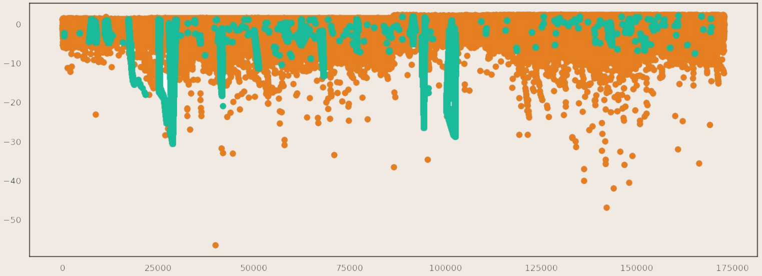
- *Borderline1-SMOTE* - for each p from S:
    1. Designate its m nearest neighbors from T. Name this set Mp, $m` = |Mp \cap L|$
    2. If $m` = m$, then p is “polluted” example. Ignore p
    3. If $0 \le m` \le m/2$, then p is “safe”. Ignore p
    4. If $m/2 \le m`$, add p to set called *DANGER*
    - For each point d from *DANGER* use the *SMOTE algorithm*
- *Bordeline2-SMOTE*
    - New points are created along lines connecting them to their closest neighbors from S or L.

\> *Borderline1-SMOTE* ($|L| = |S| = 213245$)
- *Precision = 0.25*
- *Recall = 0.73*
- *f1 = 0.37*

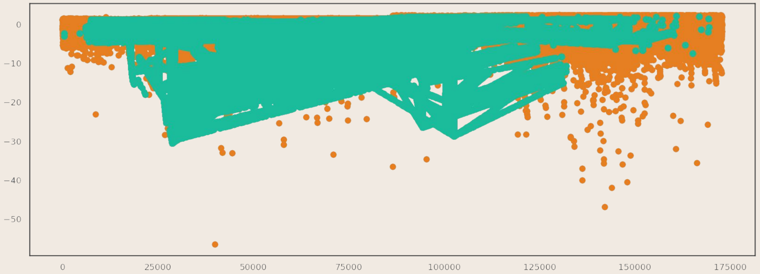
\> *Borderline2-SMOTE* ($|L| = 213245$, $|S| = 213244$)
- *Precision = 0.17*
- *Recall = 0.79*
- *f1 = 0.28*



Base dataset



Borderline1-SMOTE



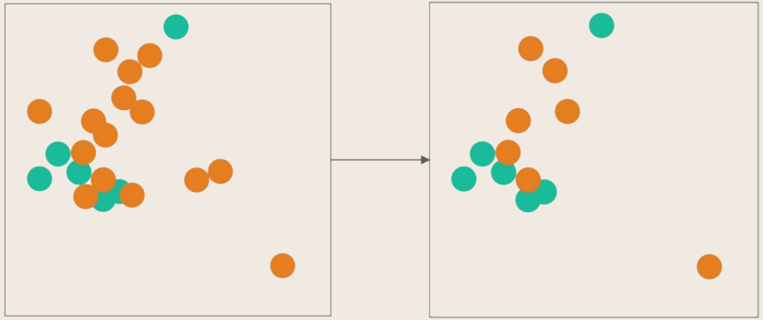
Borderline2-SMOTE

Undersampling methods

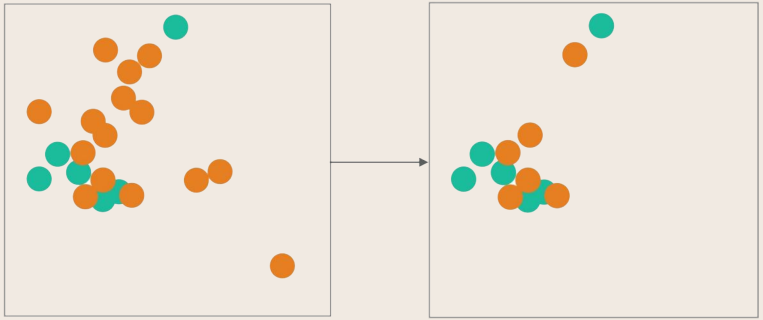
- *RUS (Random Undersampling)*
  - removal of randomly selected points from the majority class
- *Near Miss-1*
  - leaves the points from *L*, whose average distance to the *k* closest neighbors from *S* is the smallest

\> *Random Undersampling (|L| = |S| = 360)*
  - *Precision = 0.06*
  - *Recall = 0.86*
  - *f1 = 0.12*

\> *Near Miss-1 (|L| = |S| = 360)*
  - *Precision = 0.01*
  - *Recall = 0.9*
  - *f1 = 0.02*



Random undersampling



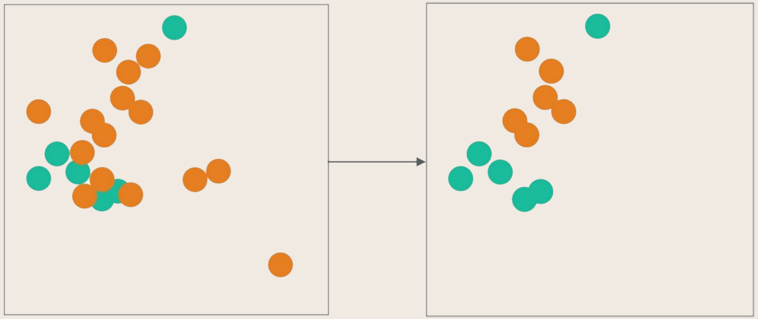
Near Miss-1, k = 3

Undersampling methods

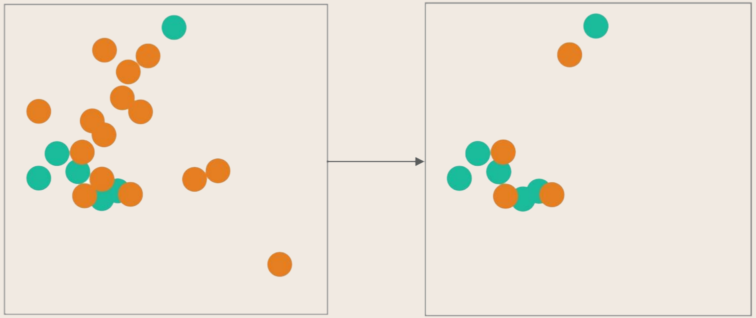
- *Near Miss-2*
  - leaves the points from *L*, whose average distance to the *k* farthest neighbors from *S* is the smallest
- *Near Miss-3*
  - chooses the *k* nearest neighbors from *L* for each point from *S*

> *Near Miss-2 (|L| = |S| = 360)*
  - *Precision = 0.02*
  - *Recall = 0.91*
  - *f1 = 0.04*

> *Near Miss-3 (|L| = |S| = 360)*
  - *Precision = 0.09*
  - *Recall = 0.88*
  - *f1 = 0.16*



Near Miss-2, k = 3



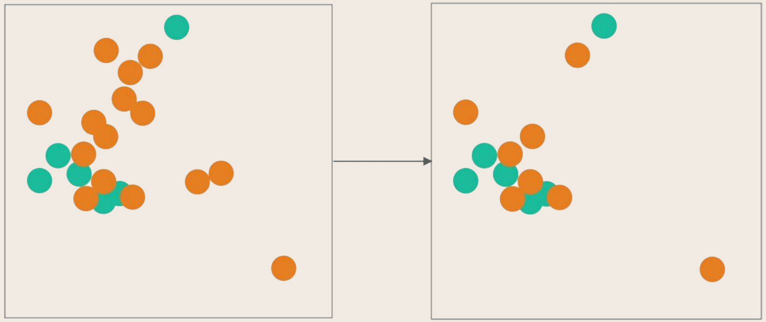
Near Miss-3, k = 3

Undersampling methods

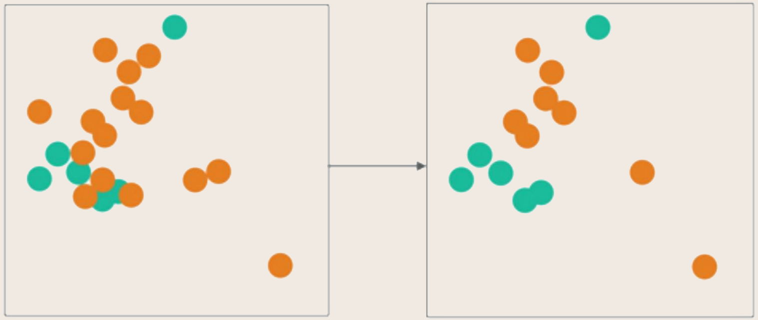
- *CNN (Condensed Nearest Neighbor)*
    1. Pick a random point *p* from *T* and set *U* = *{p}*
    2. Search *T* - *U* and add to *U* the first point you find, whose closest neighbor from *U* is of a different class
    3. Repeat step *2.* until *U* reaches its maximum
- *ENN (Edited Nearest Neighbor)*
    - remove points whose class is different from that of the majority from the *k* nearest neighbors

> *CNN (|L| = 1072, |S| = 360)*
- *Precision = 0.26*
- *Recall = 0.8*
- *f1 = 0.39*

> *ENN (|L| = 212926, |S| = 360)*
- *Precision = 0.76*
- *Recall = 0.56*
- *f1 = 0.65*



Condensed Nearest Neighbor, k = 3



Edited Nearest Neighbor

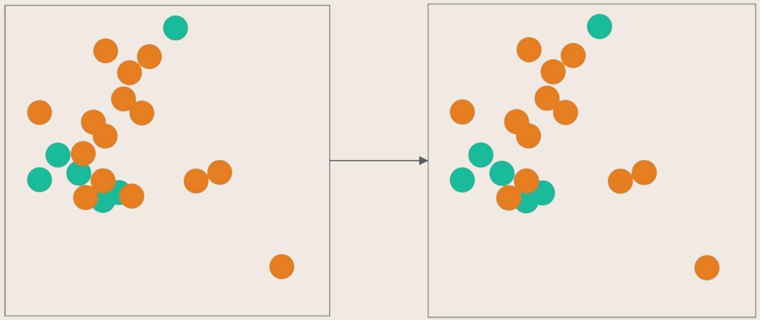
Undersampling methods

- *RENN (Repeated Edited Nearest Neighbor)*
  - repeat *ENN* until the remaining dataset is minimal
- *TLR (Tomek Link Removal)*
  - removal of all *Tomek links* from *L*

> *RENN (|L| = 212728, |S| = 360)*
  - *Precision = 0.74*
  - *Recall = 0.53*
  - *f1 = 0.62*

> *TLR (|L| = 213186, |S| =360)*
  - *Precision = 0.78*
  - *Recall = 0.58*
  - *f1 = 0.66*

Repeated Edited Nearest Neighbor



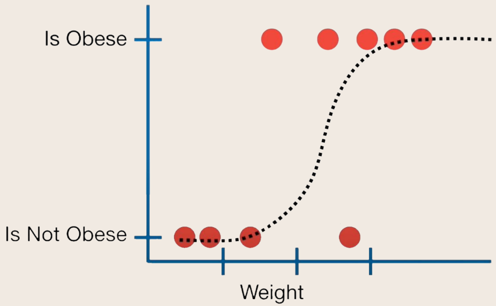
Tomek Link Removal

Examples of imbalanced data classification algorithms

- Logistic Regression
- Random Forest
- EasyEnsemble
- Support Vector Machines

Examples of imbalanced data classification algorithms

- *Logistic Regression*
  - a special case of the generalized linear model
  - the dependent variable is on a dichotomous scale (takes only two values)
  - the probability of an event occurring
- *Random Forest*
  - built on decision trees
  - uses the general bootstrap aggregation technique
  - *Weighted Random Forest*



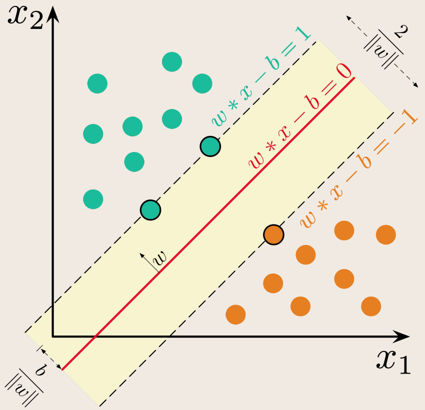
Logistic Regression

> *Logistic Regression (|L| = 71108, |S| = 94)*
  - *Precision = 0.9*
  - *Recall = 0.64*
  - *f1 = 0.75*

> *Random Forest (|L| = 71108, |S| = 94)*
  - *Precision = 0.96*
  - *Recall = 0.8*
  - *f1 = 0.87*

Examples of imbalanced data classification algorithms

- *SVM (Support Vector Machines)*
  - finds optimal hyperplanes separating extreme points
  - *maximum margin strategy*
- *EasyEnsemble* - for *i = 1, …, N:*
  1. Randomly select a subset *Li* from L such that |*Li*| = |S|
  2. Use *AdaBoost* with *Li* and *S*
  3. Combine the above classifiers into one



Support Vector Machines

> *SVM (|L| = 213245, |S| = 360)*
> - *Precision = 0.69*
> - *Recall = 0.33*
> - *f1 = 0.45*

> *EasyEnsemble (|L| = 213245, |S| = 360)*
> - *Precision = 0.05*
> - *Recall = 0.89*
> - *f1 = 0.09*