DATA MINING 2
Imbalanced Learning

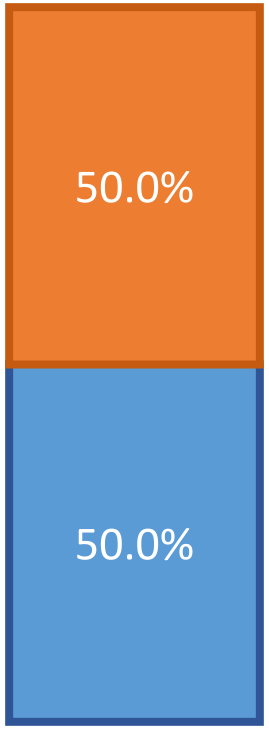
Riccardo Guidotti

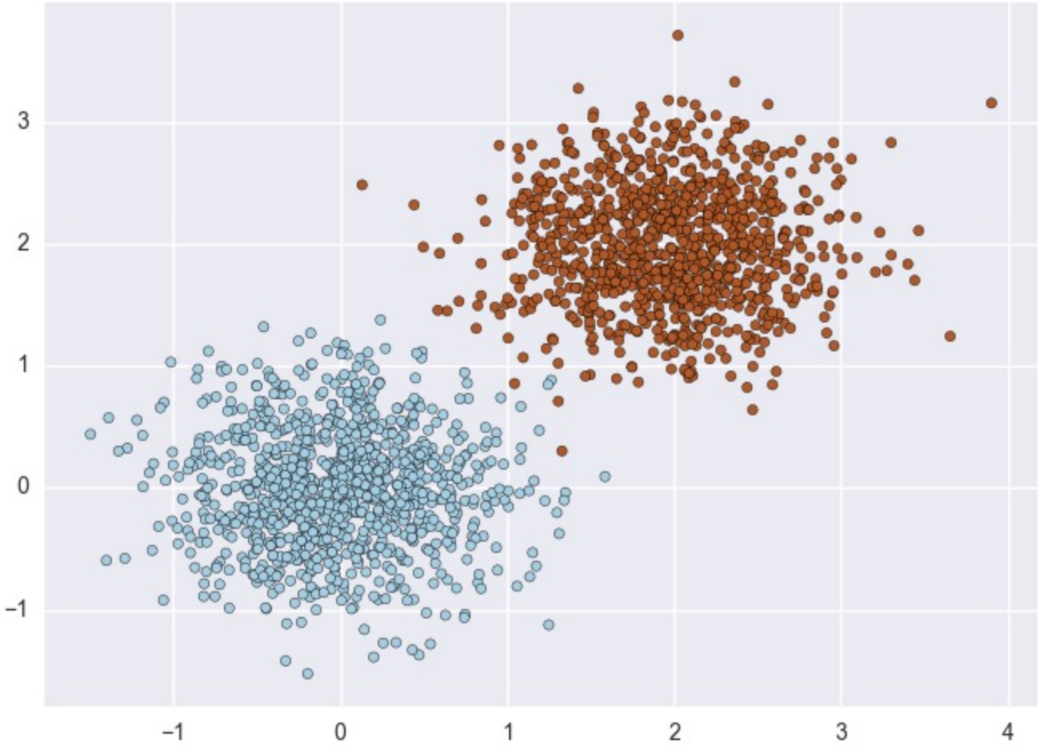
a.a. 2021/2022

UNIVERSITÀ DI PISA

Imbalanced Classes

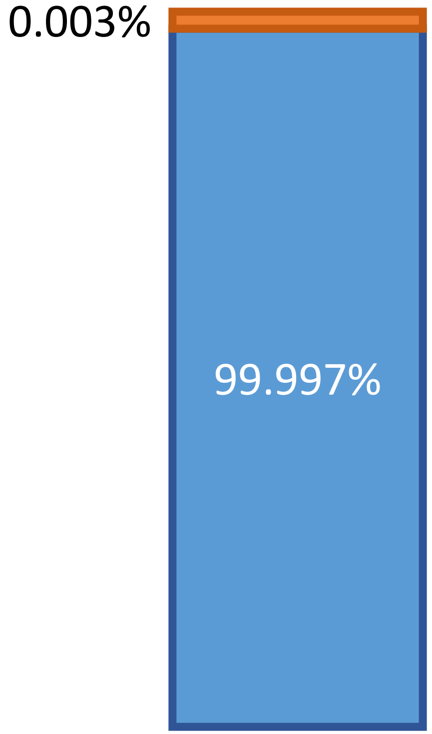
- Most classification methods assume classes are reasonably balanced.

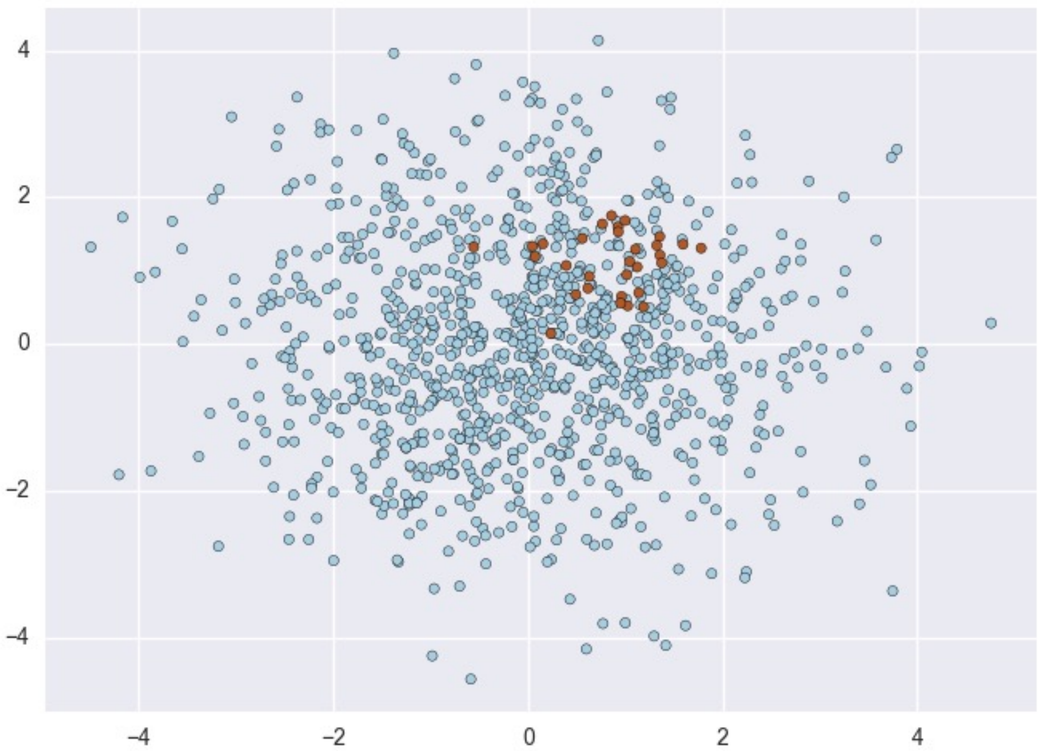




Imbalanced Classes

- In reality it is quite common to have a very popular class and a rare (yet interesting) class.





This occurs when there is a large discrepancy between the number of examples with each class label.

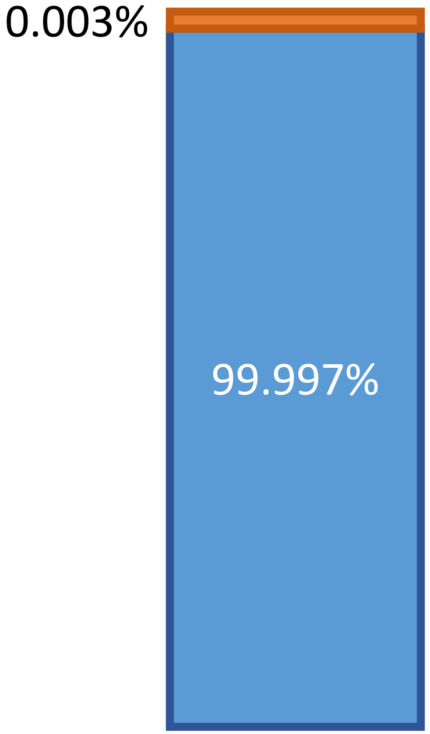
E.g. for 1M example dataset only about 30 represent an event.

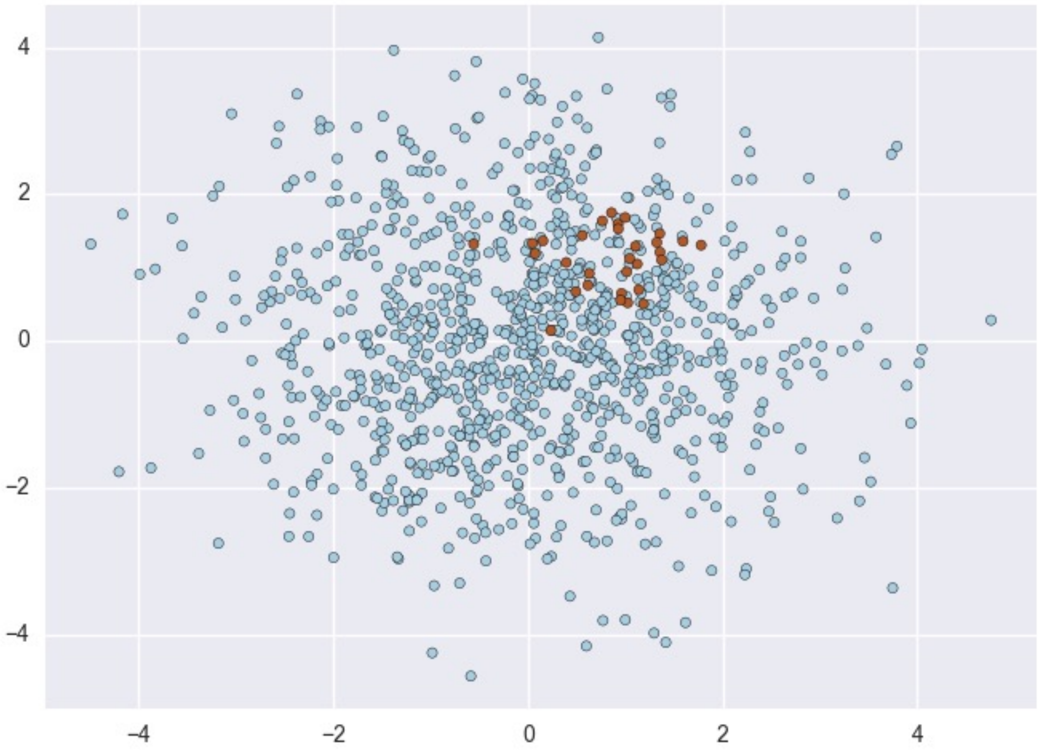
Examples

- About 2% of ***credit card*** accounts are defrauded per year1. (Most fraud detection domains are heavily imbalanced.)
- ***Medical screening*** for a condition is usually performed on a large population of people without the condition, to detect a small minority with it (e.g., HIV prevalence in the USA is ~0.4%).
- ***Disk drive failures*** are approximately ~1% per year.
- ***Factory production defect*** rates typically run about 0.1%.

What happens on classification?

- A classifier that always predict the most common class has an accuracy of 99.997%.





Evaluating Classifiers on Imbalanced Data

- When classes are slightly imbalanced, no balancing is need.
- Yet, take that into consideration when evaluating performances
- Assume the test set contains 100 records

  - Positive cases = 75, Negative cases = 25
  - Is a classifier with 70% accuracy good?
  - No, the trivial classifier (always positive) reaches 75%

  - Positive cases = 50, Negative cases = 50
  - Is a classifier with 70% accuracy good?
  - At least much better than the trivial classifier

Multiclass Problem

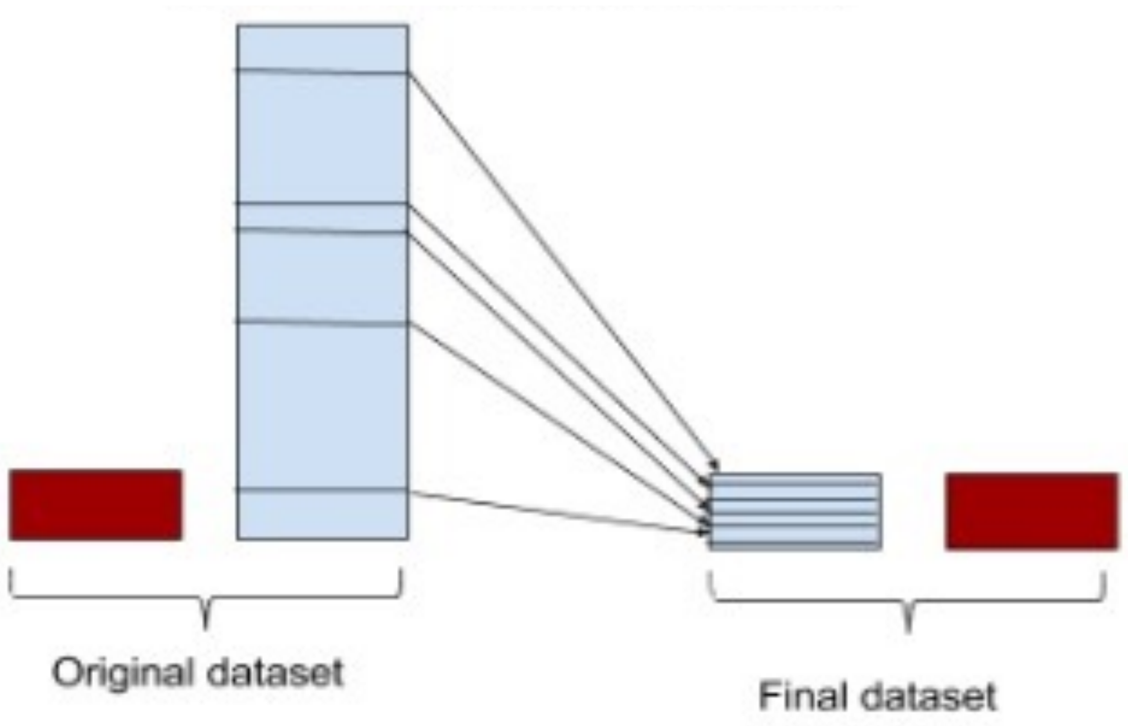
- Assume N classes
- If classes are perfectly balanced, a trivial classifier (e.g. majority) will yield $A_{trivial}$ ~100/N % accuracy
- N=2 → $A_{trivial}$ ~ 50%
- N=4 → $A_{trivial}$ ~ 25%
- Goodness of accuracy of a model should be compared against $A_{trivial}$
- E.g., If N=5, an accuracy of 40% would look large

Handling Imbalanced Data

- Balance the training set
  - Undersampling the majority class
  - Oversampling the minority class
- At the algorithm level
  - Adjust the class weight by making the algorithm more sensitive to rare classes
  - Adjust the decision threshold
  - Design new algorithm to perform well on imbalanced data
- Switch to anomaly detection
- Do nothing and hope to be lucky

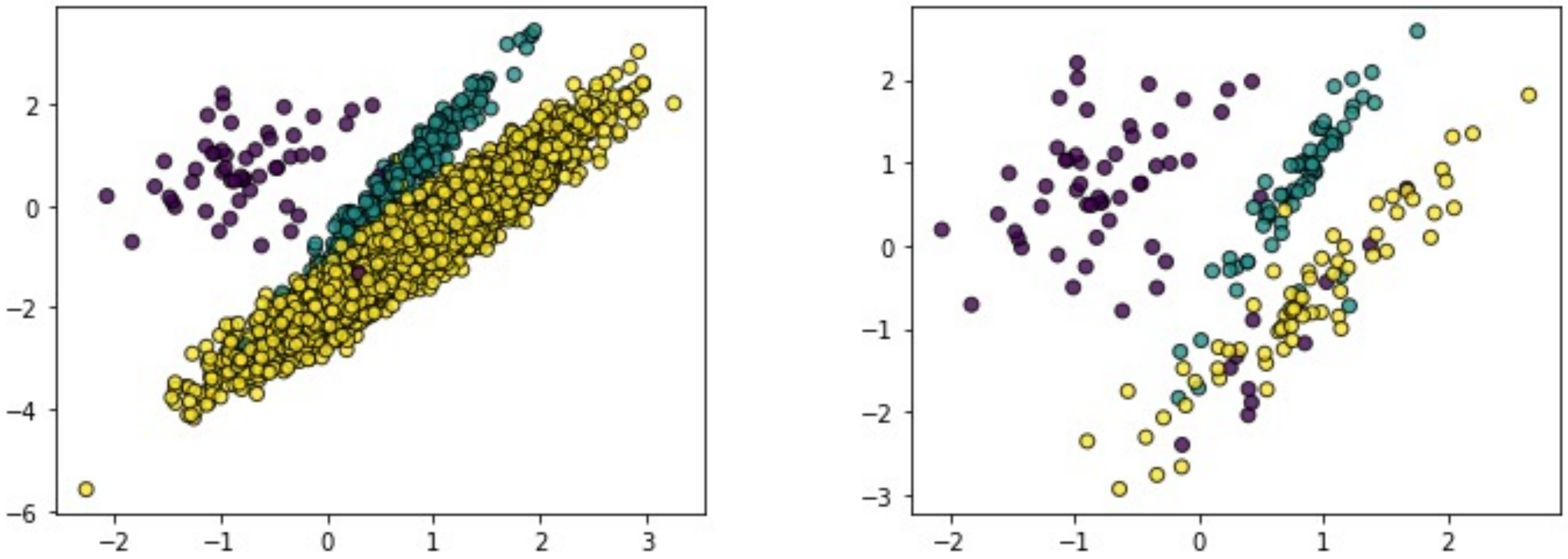
Undersampling the Majority Class

- Random Undersampling
- Neighbor-based approaches, e.g., Condensed Nearest Neighbor, Tomek Links, etc.



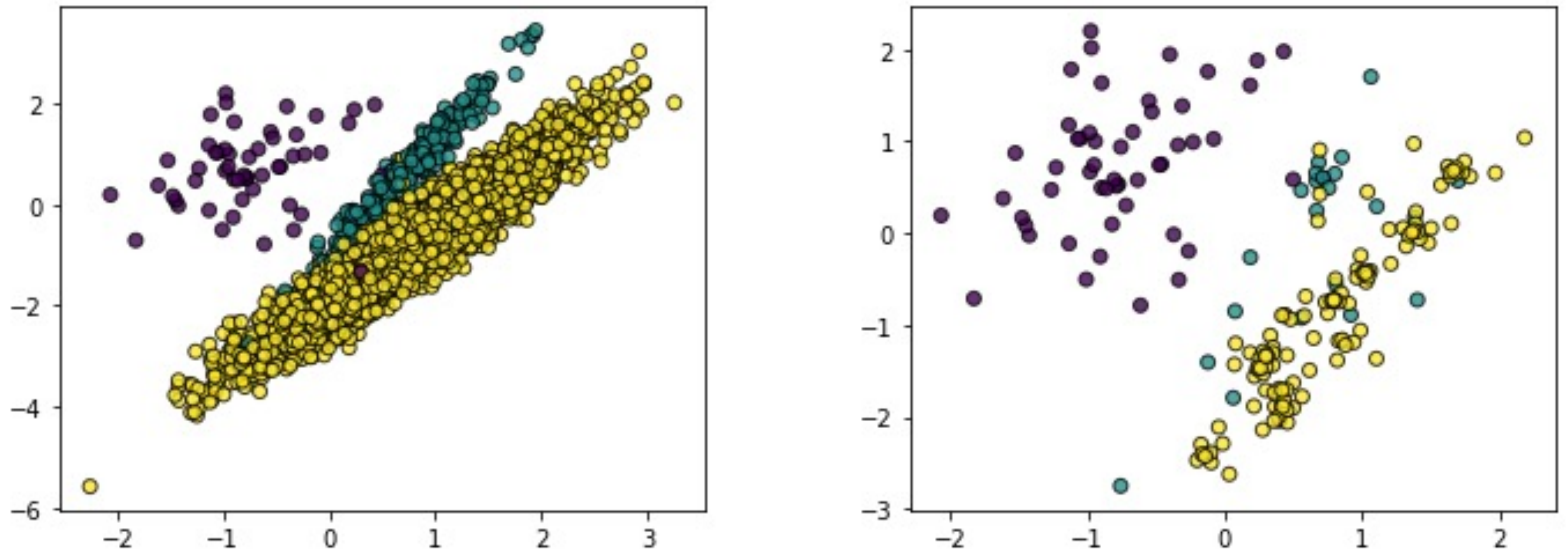
Random Undersampling

- Under-sample the majority class(es) by randomly picking samples with or without replacement.



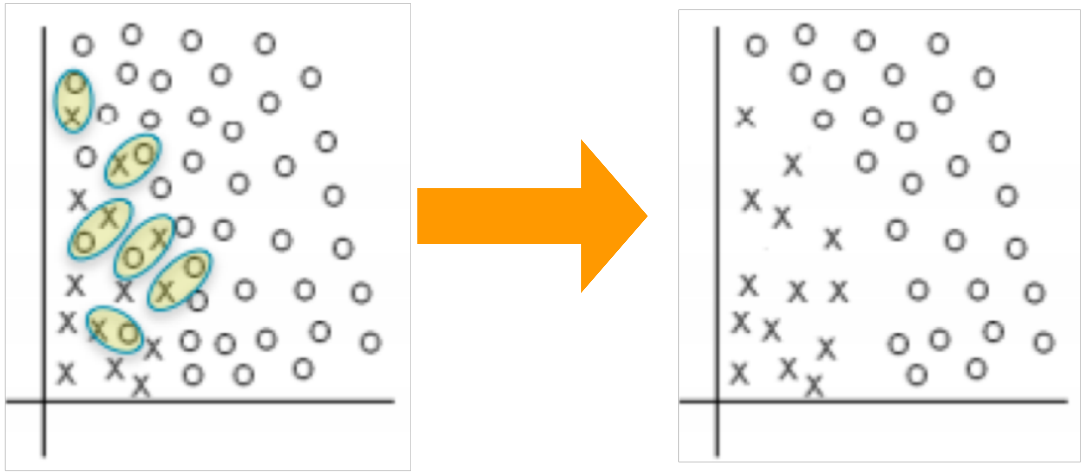
Condensed Nearest Neighbor

- Perform a smart undersampling by removing majority points having as k-NN a minority point.



Condensed Nearest Neighbor

P. Hart, “The condensed nearest neighbor rule,” In Information Theory, IEEE Transactions on, vol. 14(3), pp. 515-516, 1968



1) The first sample is placed in STORE.

2) The second sample is classified by the NN rule, using as a reference set the current contents of STORE. (Since STORE has only one point, the classification is trivial at this stage.) If the second sample is classified correctly it is placed in GRABBAG; otherwise it is placed in STORE.

3) Proceeding inductively, the ith sample is classified by the current contents of STORE. If classified correctly it is placed in GRABBAG; otherwise it is placed in STORE.

4) After one pass through the original sample set, the procedure continues to loop through GRABBAG until termination, which can occur in one of two ways:

a) The GRABBAG is exhausted, with all its members now transferred to STORE (in which case, the consistent subset found is the entire original set), or

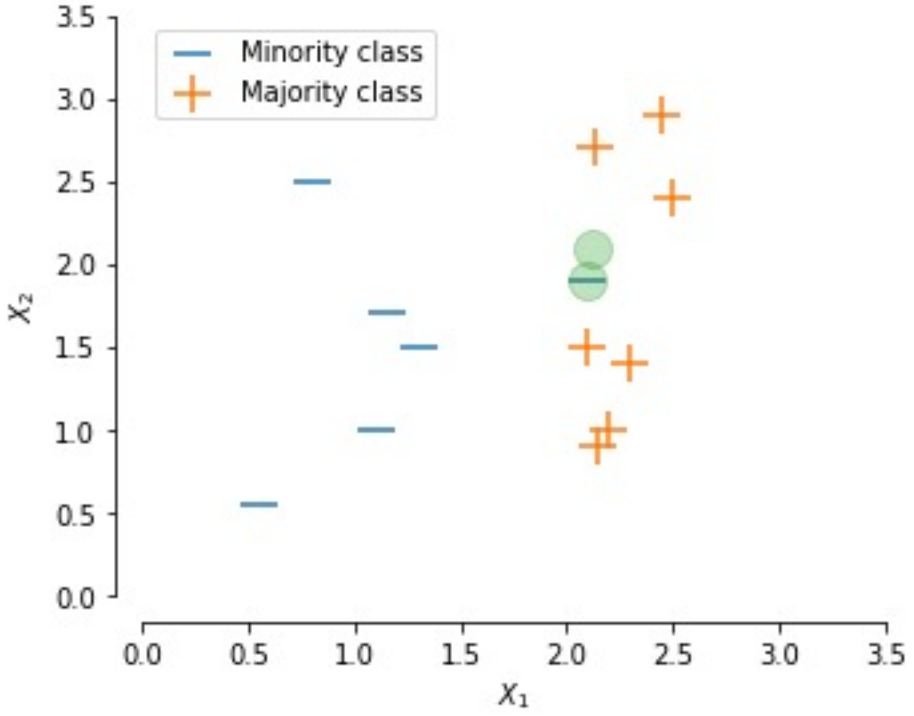
b) One complete pass is made through GRABBAG with no transfers to STORE. (If this happens, all subsequent passes through GRABBAG will result in no transfers, since the underlying decision surface has not been changed.)

5) The final contents of STORE are used as reference points for the NN rule; the contents of GRABBAG are discarded.

Condensed Nearest Neighbor

grabbag store

a) pass ← 1,
b) choose $x \in D$ randomly, $D(1) = D - \{x\}$, $E = \{x\}$,
c) $D(\text{pass} + 1) = \varnothing$, count ← 0,
d) choose $x \in D(\text{pass})$ randomly, classify x by NN using E,
e) if classification found in d) agrees with actual membership of x
   then $D(\text{pass} + 1) = D(\text{pass} + 1) \cup \{x\}$
   else $E = E \cup \{x\}$, count ← count + 1,
f) $D(\text{pass}) = D(\text{pass}) - \{x\}$,
g) if $D(\text{pass}) \neq \varnothing$ go to d),
h) if count = 0
   then end of algorithm
   else pass ← pass + 1, go to b).

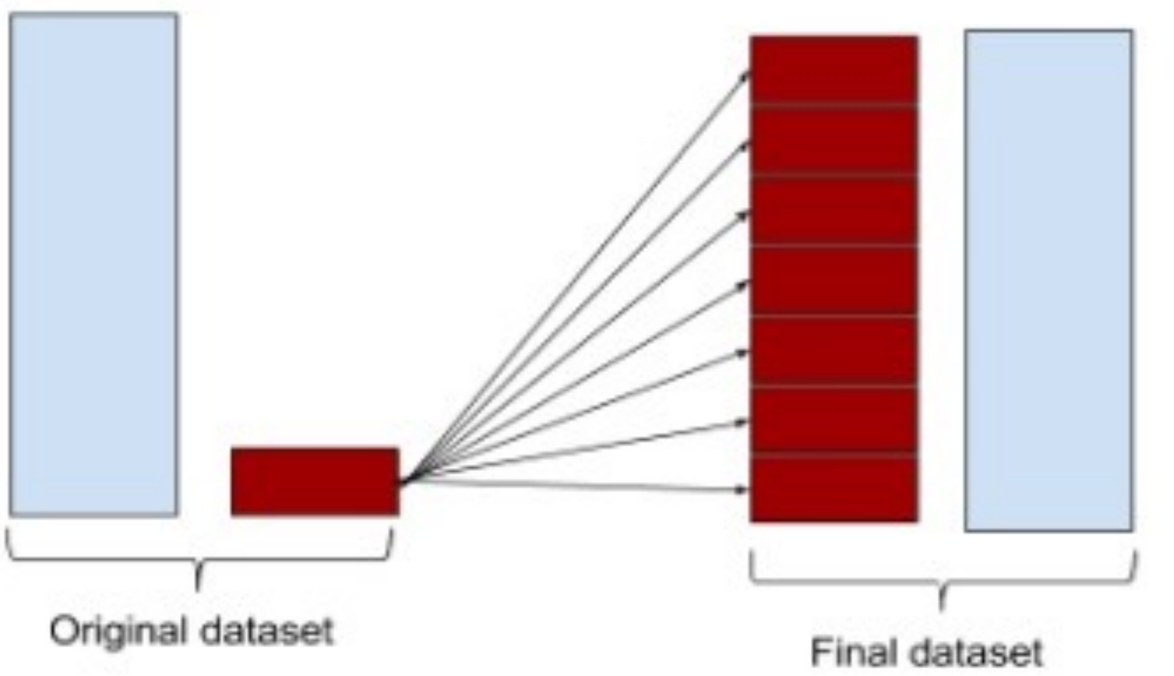


CNN alternatives

- Tomek’s links
- One Sided Selection

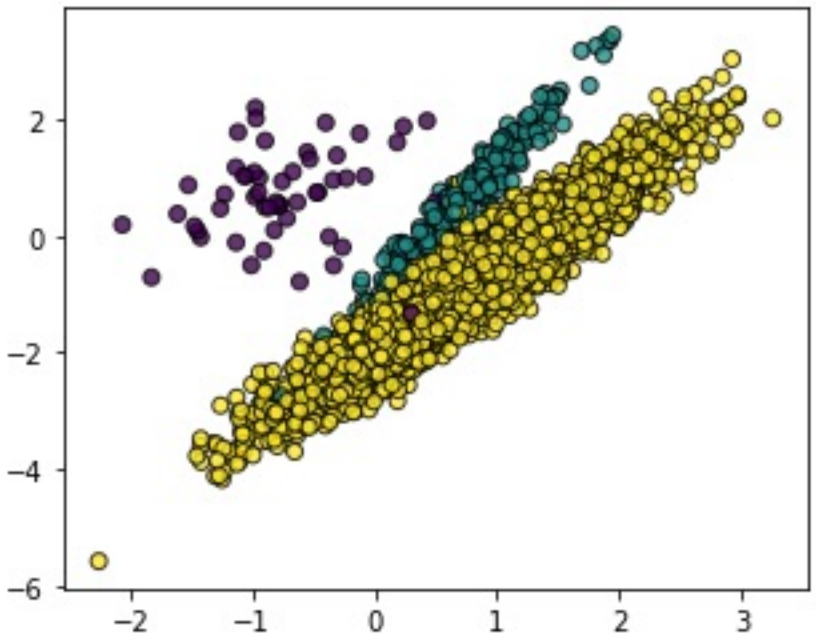
Oversampling the Majority Class

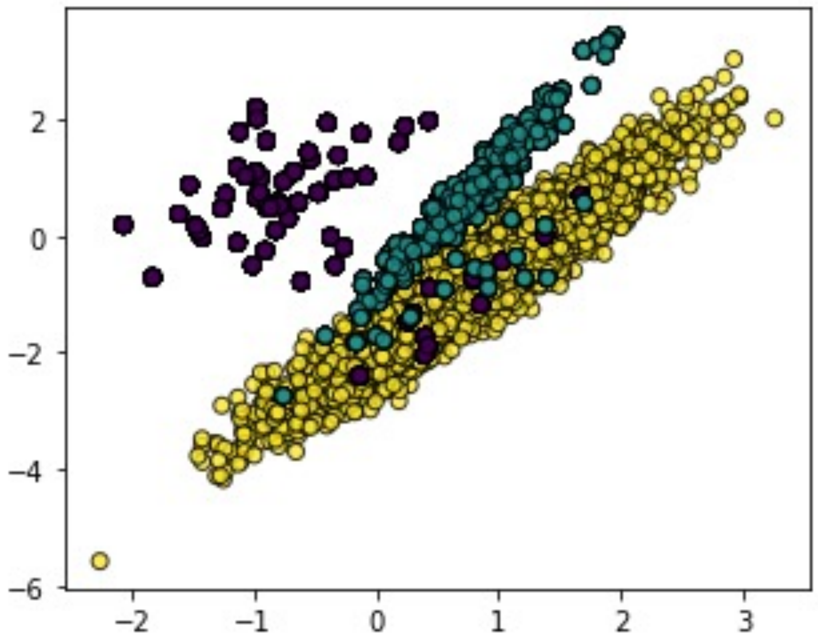
- Random Oversampling
- Synthetic Minority Oversampling Technique (SMOTE)
- Adaptive Synthetic (ADASYN) sampling approach



Random Oversampling

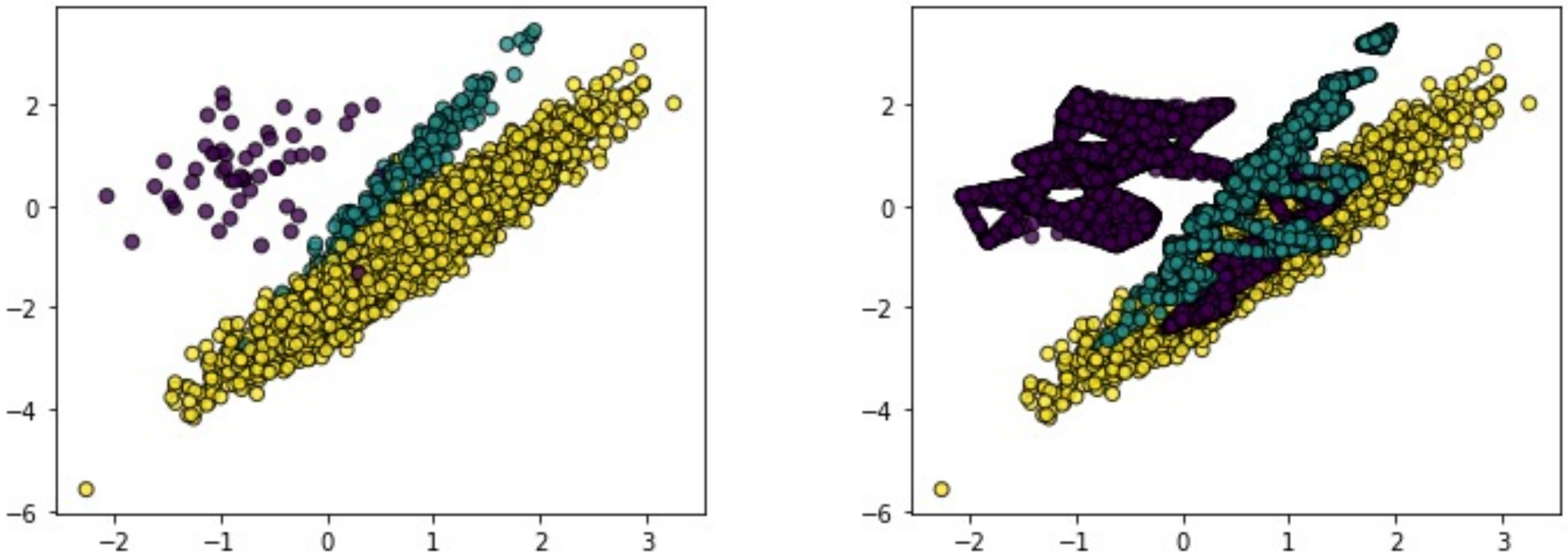
- Over-sample the minority class(es) by picking samples at random with replacement.





SMOTE Oversampling

- Over-sample the minority class(es) by adding points through interpolation.

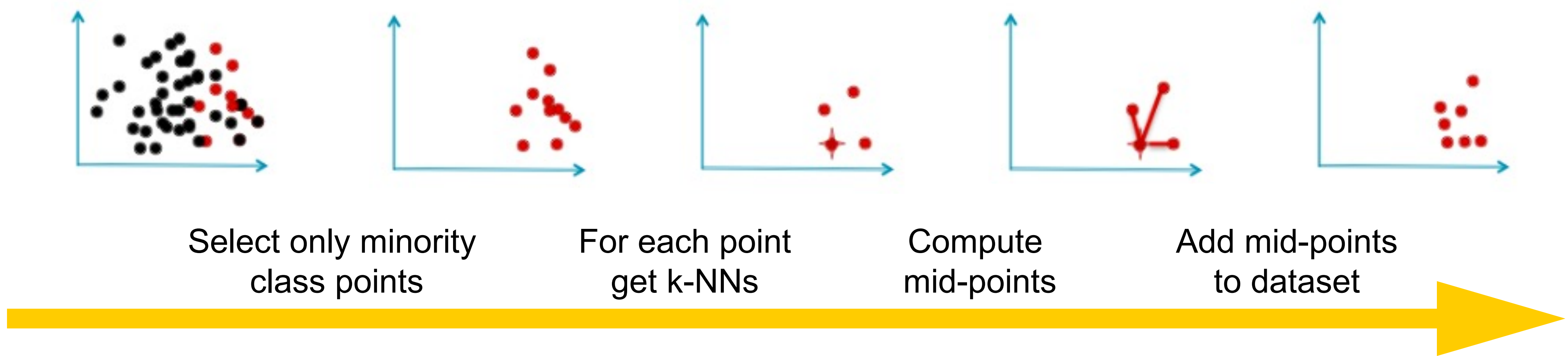


SMOTE

- It operates in the ***"feature space"*** rather than in the "data space", and effectively forces the decision region of the minority class to become more general.
- The minority class is over-sampled by taking each minority class sample and ***introducing synthetic examples along the line segments joining any/all of the k minority class nearest neighbors***.
- Depending upon the amount of over-sampling required, ***neighbors from the k nearest neighbors are randomly chosen*** (by default k=5).
- E.g., if the amount of over-sampling needed is 200%, only two neighbors from the five are chosen and one sample is generated in the direction of each.

SMOTE – Samples Generation

- Take the difference between the feature vector (sample) under consideration and its nearest neighbor.
- Multiply this difference by a random number between 0 and 1, and add it to the feature vector under consideration.
- This causes the selection of a random point along the line segment between two specific features.

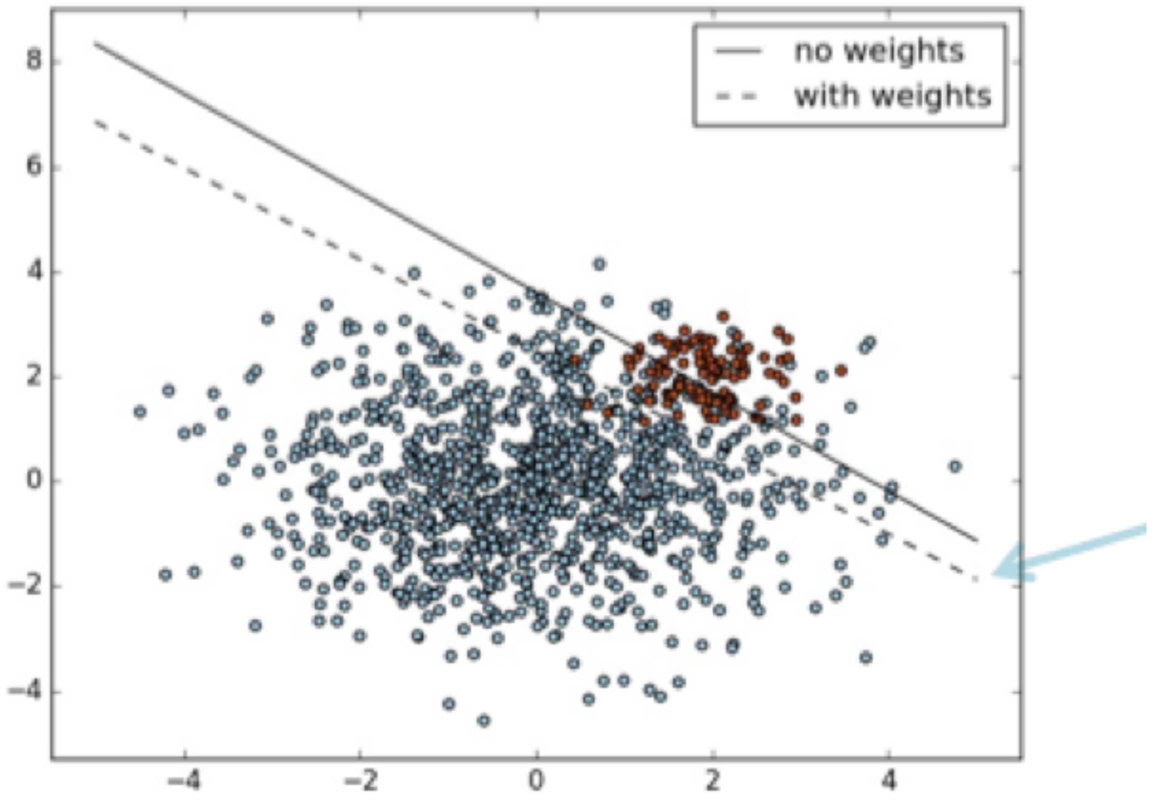


SMOTE alternatives

- SMOTENC: Over-sample for continuous and categorical features.
- BorderlineSMOTE: Over-sample using the borderline variant.
- SVMSMOTE: Over-sample using the SVM variant.
- ADASYN: Over-sample using ADASYN.

Adjust the Class Weight

- The classifier can be trained considering **different costs** to be paid for misclassification errors on minority classes.
- This is generally done using a “class weight’’.



Adjust the Class Weight

- Each outcome with respect to a confusion matrix can be associated to a weight in a corresponding weight (or cost) matrix.
- Thus, the objective of the classification algorithm is to find the model that minimizes the total cost.
  - $\sum_X weight(x)freq(x)$

Confusion Matrix

	Actual	
Predicted	Y	N
Y	50	7
N	3	40

Weight Matrix

	Actual	
Predicted	Y	N
Y	0	95
N	5	0

Cost = 0.03*5 + 0.07*95

Meta-Cost Sensitive Classifier

- Apply a classifier getting probability of a class label *P(j|x)*
- Compute expected risk of classifying *x* with class *i*:

$$R(i|x) = \sum_j P(j|x)C(i,j)$$

- Re-label the train data with the class i having lower risk
- Learn a model on the cost-sensitive train data

Adjust the Decision Threshold

- Several classification methods compute scores in terms of probability of belonging to a class, and then assign class.
- Generally we have:
  - Score $p > 50\%$ → class = Y
  - Otherwise → class = N

- E.g.: decision trees have p = #positive/#negative cases over each leaf

Adjust the Decision Threshold

- What if we generalize the schema into:
  - Score p > THR% → class = Y
  - Otherwise → class = N
- For each THR (in [0-100]) we get a different set of predictions
- The confusion matrix changes and all indicators derived from it change
  - Accuracy
  - True Positive Rate (TPR)
  - False Positive Rate (FPR)
  - ….

https://www.wooclap.com/DM2IMBLEARN

References

- I. Tomek, “Two modifications of CNN,” In Systems, Man, and Cybernetics, IEEE Transactions on, vol. 6, pp 769-772, 2010.
- N. V. Chawla, K. W. Bowyer, L. O.Hall, W. P. Kegelmeyer, “SMOTE: synthetic minority over-sampling technique,” Journal of artificial intelligence research, 321-357, 2002.
- Domingos, Pedro. "Metacost: A general method for making classifiers cost-sensitive." Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining. 1999.
- P. Hart, “The condensed nearest neighbor rule,” In Information Theory, IEEE Transactions on, vol. 14(3), pp. 515-516, 1968.
- Python *imblearn* library: https://imbalanced-learn.readthedocs.io/en/stable/index.html