# Undersampling methods – Introduction

1. Undersampling methods

* Reducing the number of observations from the majority class

1. Fixed vs. cleaning undersampling

* Fixed: reduce the majority class to the same number of observations as the minority
* Random
* NearMiss
* Instance Hardness
* Cleaning: clean the majority class based on some criteria
* All other methods

1. Balancing ratio

* Fixed undersampling
* Remove samples from the majority class until
* But user could determine otherwise, for example , that is twice as many from the majority class as those from the minority

1. Under sampling criteria

A screenshot of a computer

Description automatically generated

1. Remove noisy observations

A diagram of a class

Description automatically generated

1. Retain closer observations

A diagram of a class

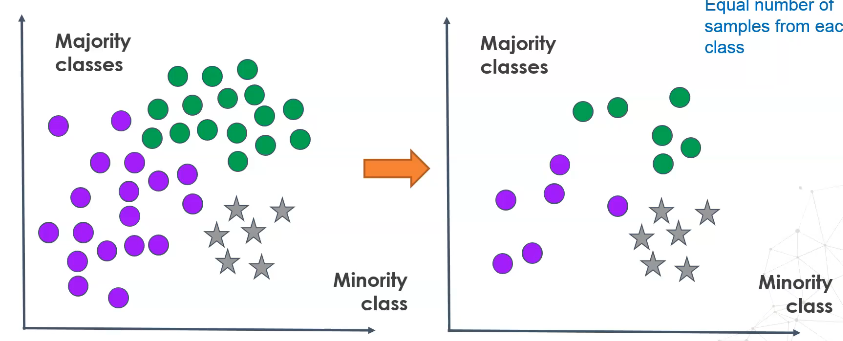
Description automatically generated

# Random undersampling

1. Random undersampling

* Extracts observations at random from the majority class until a certain balancing ratio is reached
* Naïve technique

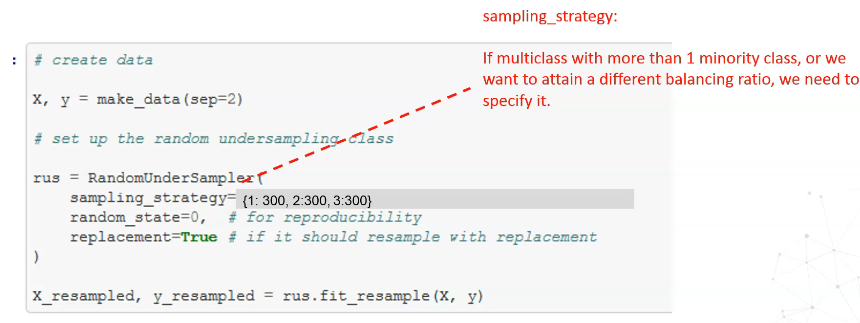
1. Multiclass



1. Imbalanced-learn: RandomUnderSampler

A screen shot of a computer

Description automatically generated



* Replacement: If True, the same observation can be sampled more than once. In general False, unless we have very few observations

1. Considerations

A diagram of a model

Description automatically generated with medium confidence

# Condensed nearest neighbors

1. Condensed nearest neighbors (CNN)

* Extracts observations at the boundary between the 2 or more classes
* Cleaning
* Final dataset shape varies
* Boundary matters
* Step 1: separate minority class into a group
* Step 2: take 1 random observation from majority class and move it to minority class
* Step 3: train a 1 KNN algorithm
* Step 4: use KNN algorithm to classify observations from majority class one at a time. If the prediction matches the real class, exclude the sample and evaluate another observation. If the prediction does not match real class, pass it to minority group.
* Step 5: train a new KNN algorithm
* Repeat until all observations from majority class have been evaluated
* The final dataset contains the minority class + all observations from the majority class that were wrongly classified by the subsequent KNN algorithms

1. Considerations

* Pros: Focus on harder cases -> improves performance
* Cons: introduces noise

1. Imbalanced-learn implementation

A computer code with text

Description automatically generated with medium confidence

1. Multi-class

* One vs. one
* Run entire procedure over 1 majority class first
* Repeat the procedure for the other majority classes
* Disadvantage: does not scale very well

# Tomek Links

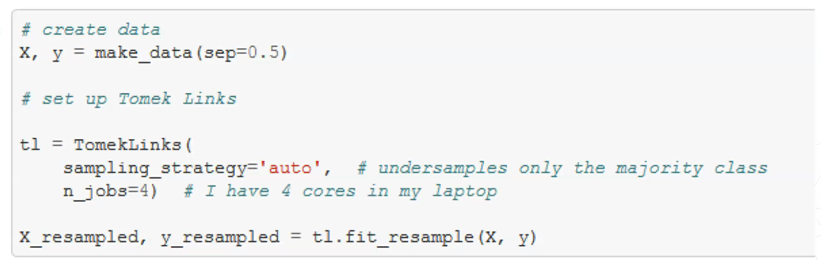
1. Tomek Links

* If 2 samples are nearest neighbors and from a different class, they are Tomek Links
* Removes the Tomek Link from the majority class
* Cleaning
* Final dataset shape varies
* Boundary is noise

1. Considerations

* Removes noise -> improves performance
* Misclassifies the hard cases

1. Imbalanced-learn implementation



1. Multiclass

* One vs. rest
* Remove sample from majority or remove entire link

# One Sided Selection

1. One sided selection

* Retain observations from the majority that are hard to classify but remove the noise
* First, selects the samples at the boundary of the classes (hardest instances)
* Next, removes Tomek links
* Cleaning
* Final data shape varies
* Boundary matters

1. Procedure

* Step 1: create group S with all samples from minority class
* Step 2: add 1 observation from the majority to S
* Step 3: train a 1 KNN on S
* Step 4: make predictions on the rest of the majority class obs
* Step 5: if predictions don’t match the class, pass the samples to S
* Step 6: in S, find and remove Tomek Links

1. Imbalanced-learn implementation
2. Multiclass

* One vs. one

# Edited Nearest Neighbours

1. Edited Nearest Neighbours

* Remove samples from the majority class that are closest to the boundary (with the other classes)
* Enhance the separation of the classes
* Removes the observations whose neighbours disagree with it on the class
* Typically, 3 neighbours per observation are evaluated
* Final dataset shape varies
* Cleaning
* Removes hard cases

1. Procedure

* Trains a 3 KNN on entire dataset
* Finds each observation’s 3 closest neighbours
* Keeps or removes observations based on neighbours agreement with its class. 2 selection criteria:
* All neighbours need to agree to retain observation
* Most neighbours need to agree to retain observation

1. Implementation

A screenshot of a computer

Description automatically generated

1. Multiclass

* One vs. rest
* Only majority classes are undersampled
* When all or most neighbours agree, the observation is retained

# Repeated Edited Nearest Neighbours

1. Repeated Edited Nearest Neighbours

* Trains 3 KNN on entire dataset
* Finds each observation’s 3 closest neighbours
* Decides whether to keep or remove based on neighbours agreement with its class
* **Repeats 1 to 3**
* Until no more observations are removed or,
* A maximum number of cycles is reached
* Final dataset shape varies
* Cleaning
* Removes hard cases
* Various passes over the dataset
* Always builds KNN with same number of neighbours

1. Implementation - RENN

A computer code with text

Description automatically generated with medium confidence

1. Multiclass

* One vs Rest

# All K Nearest Neighbours

1. All KNN

* Remove samples from the majority class that are closest to the boundary
* Repeats ENN
* Starts by exploring the 1 closest neighbour
* Adds 1 neighbour to the KNN at each round
* Stops after examining a user defined maximum number of neighbours
* Or when the majority class becomes minority class

1. Procedure

* Trains 1 KNN on entire dataset
* Finds each obs’ 1 closest neighbour
* Decides whether to keep or remove, based on neighbours agreement with its class
* Repeats but adding 1 K to the KNN until
* A max number of neighbours is examined
* Or when the majority class becomes the minority class
* Final dataset shape varies
* Cleaning
* Removes hard cases
* Removes more samples than ENN
* Various passes over the dataset
* Successive KNNs have more neighbours
* More observations need to agree on the class

1. Implementation

A screen shot of a computer

Description automatically generated

1. Multiclass

* One vs. rest
* Only majority classes are undersampled
* When all or most neighbours agree, the observation is retained

# Neighbourhood Cleaning Rule

1. Neighbourhood Cleaning Rule

* Remove samples from the majority class that are closest to the boundary (with the other classes)
* Expands on ENN, by cleaning examples from the majority class that are neighbours to the minority
* Enhance the separation of the classes, remove noise
* Cleaning
* Final dataset shape varies
* Removes hard cases

1. Procedure

* Step 1
* Trains a 3 KNN on entire dataset
* Finds each observation’s 3 closest neighbours (for majority classes only)
* Keeps or removes ob based on neighbours agreement with its class
* So far, ENN
* Step 2: Now clean further
* Find the 3 neighbours of each observation from the minority class
* If all or more neighbours disagree with the minority class, remove them (it removes all neighbours that disagree)
* Except: if the neighbours belong to a class with few samples. In the original article, they would only remove a neighbour if it belongs to a class with at least half as many observations as those in the minority

1. Implementation

A computer screen shot of a computer code

Description automatically generated

1. Multiclass

* One vs. rest
* Only majority classes are removed
* When most neighbours disagree, flag the observation
* When all or most neighbours disagree, flag the observation
* Final dataset = original minus flagged observations

# NearMiss

1. NearMiss

* 3 versions
* Fixed method
* Final dataset is 2 x minority for binary classification
* Retains info closer to the minority class
* Retains info closer to the minority class
* Design to work with text, where each word is a complex representation of words and tags

1. NearMiss – version 1

* Determine the mean distance to each k closest neighbour from X(min)
* Retain observations from X(maj) with the smallest average distance

1. NearMiss – version 2

* Determine the mean distance to each k furthest neighbour from X(min)
* Retain observations from X(maj) with the smallest average distance

1. NearMiss – version 3

* Retain the 3 closest K to the minority sample -> Intermediate dataset
* Select those which average distance to X(min) is the largest

1. Implementation

A computer code on a white background

Description automatically generated

1. Multiclass

* One vs. rest

# Instance hardness threshold

1. Definition

* Instance hardness – a measure of how difficult it is to classify an instance or observation correctly
* Hard instances are observations that are hard to classify correctly
* Class overlap is the principal contributor to instance hardness
* Instance hardness – the probability of misclassification of an observation – depends on:
* The learning algorithm used to model the task
* The observation’s relation to other observations (class overlap)
* Some instances are harder for some algorithms than others
* Fundamentally, instances that are hard to classify correctly are those for which the learning algorithm has a low probability of predicting the correct class label
* Instance hardness = 1 – probability
* Hard instances
* High metric
* Class probability low

1. Instance hardness filtering

* Remove hard instances from data to reduce class overlap -> increase class separation
* Remove instances with an instance hardness greater than a threshold
* Determine the threshold arbitrarily
* Find threshold to match a desired balancing ratio, like imbalanced-learn
* To select as many observations from the majority, as those from the minority -> use percentile
* If desired #obs is 10 and #obs majority class is 90 -> perc = (1 – 10/90) x 100 = 88.89

1. Procedure

* Train a ML algorithm
* Determine the instance hardness
* Remove observations with high instance hardness (or equivalently, with low prob of class)
* If more than 2 classes -> 1 vs rest approach to determine hardness

1. Considerations

* Filter with the same algorithm that you intend to train
* The beauty of instance hardness is that various thresholds can be used and compared
* Instance hardness filtering was designed to improve classifier performance in general, not just for imbalanced datasets.