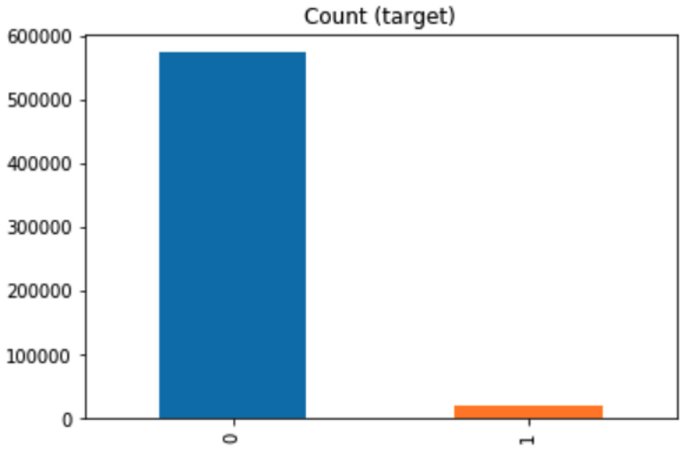
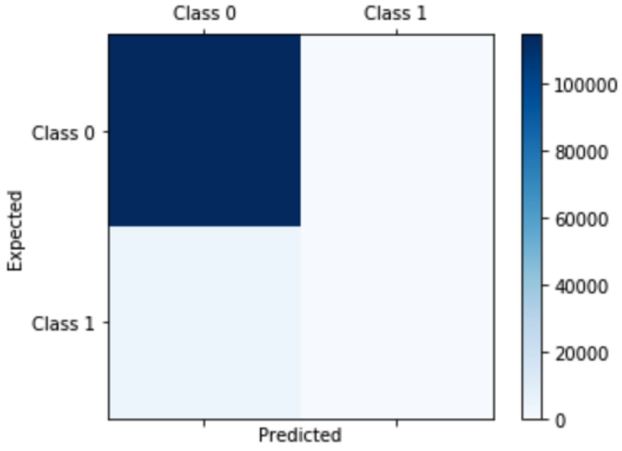
Case study: Imbalanced data

IKMLab

Imbalanced data



A learned classifier



0: majority class
1: minority class

Accuracy: 96.36%

Reference: https://www.kaggle.com/rafjaa/resampling-strategies-for-imbalanced-datasets

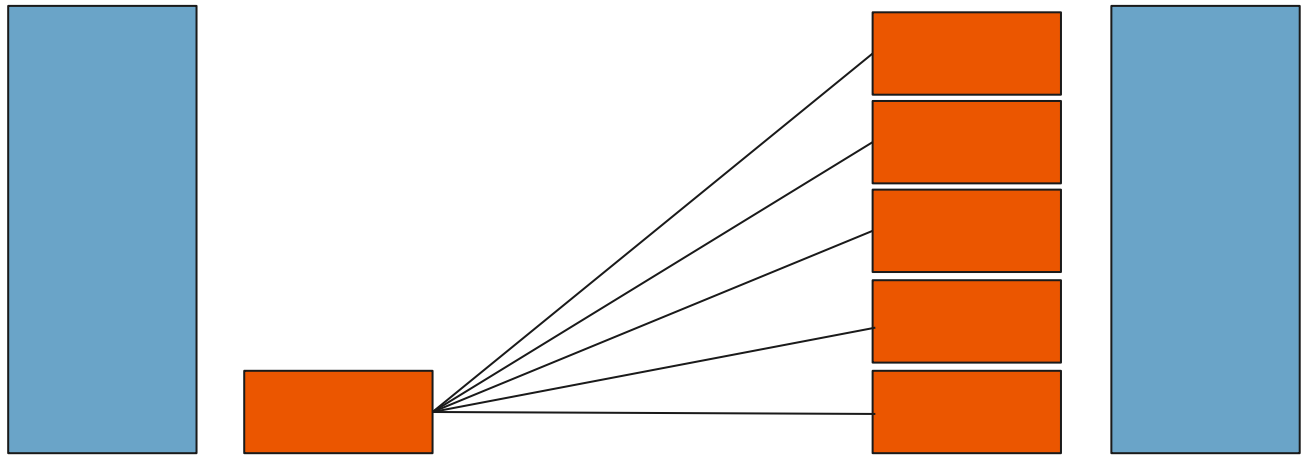
少的變多ex: 全部複製1000筆（不太好）、資料擴增ex: 文字->翻譯成法文再翻回中文，句子多了一句
多的變少

Solutions to imbalanced data

- Up-sampling (over-sampling)
  - Create more data of the minority classes
- Down-sampling (Under-sampling)
  - Reduce some data of the majority classes
- Cost-sensitive learning
  - Weight the minority classes and the majority classes to treat them differently during training

Up-sampling

Upsampling or **Oversampling** refers to the technique to create artificial or duplicate data points or of the minority class sample to balance the class labels. Below is upsampling Illustration.



Original dataset

Up-sampling techniques

- **Random Over Sample:**

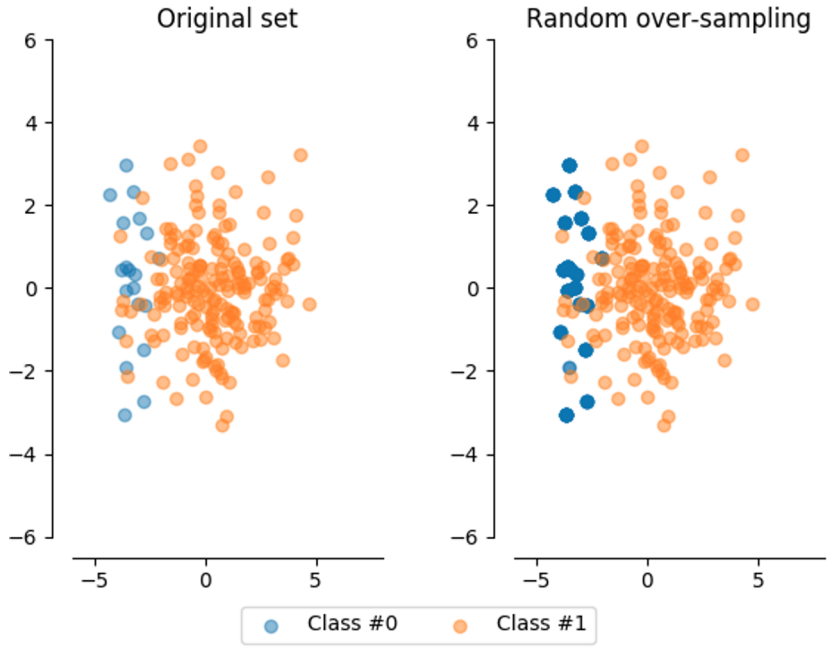
  Random oversampling is the simplest oversampling technique to balance the imbalanced nature of the dataset.

- **SMOTE (**Synthetic Minority Oversampling Technique**):**

  It creates new synthetic samples to balance the dataset.

Random Over Sample

- **Random oversampling** : Randomly duplicate examples in the minority class.
- It balances the data by replicating the minority class samples. This does not cause any loss of information, but the dataset is prone to **overfitting** as the same information is copied.



Reference: http://glemaitre.github.io/imbalanced-learn/auto_examples/over-sampling/plot_random_over_sampling.html

SMOTE (Synthetic Minority Oversampling Technique)

- Random oversampling just increases the size of the training data set through repetition of the original examples. It does not cause any increase in the variety of training examples.

- Oversampling using **SMOTE** not only increases the size of the training data set, it also **increases the variety**.

- SMOTE, when done right, is preferable over random oversampling.

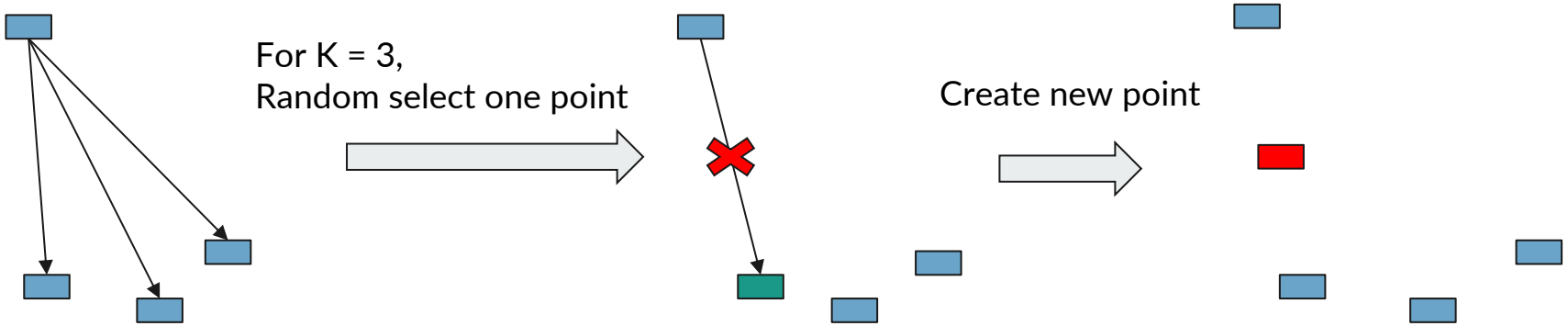
SMOTE (Synthetic Minority Oversampling Technique)

SMOTE works by utilizing a **k-nearest neighbor** algorithm to create synthetic data. Steps samples are created using Smote:

1. For one minority class **X**,find its **K** nearest neighbors.
2. Randomly choose one neighbor and compute the distance called **dif** between minority class and its nearest neighbor.
3. Choose a random number called **gap** between 0 and 1.
4. New sample X' = X+ **dif** * **gap**
5. Repeat the process for resampling the dataset.

Paper: SMOTE: Synthetic Minority Over-sampling Technique, Journal of Artificial Intelligence Research (2002).

SMOTE (Synthetic Minority Oversampling Technique)



Synthetic Minority Oversampling Technique

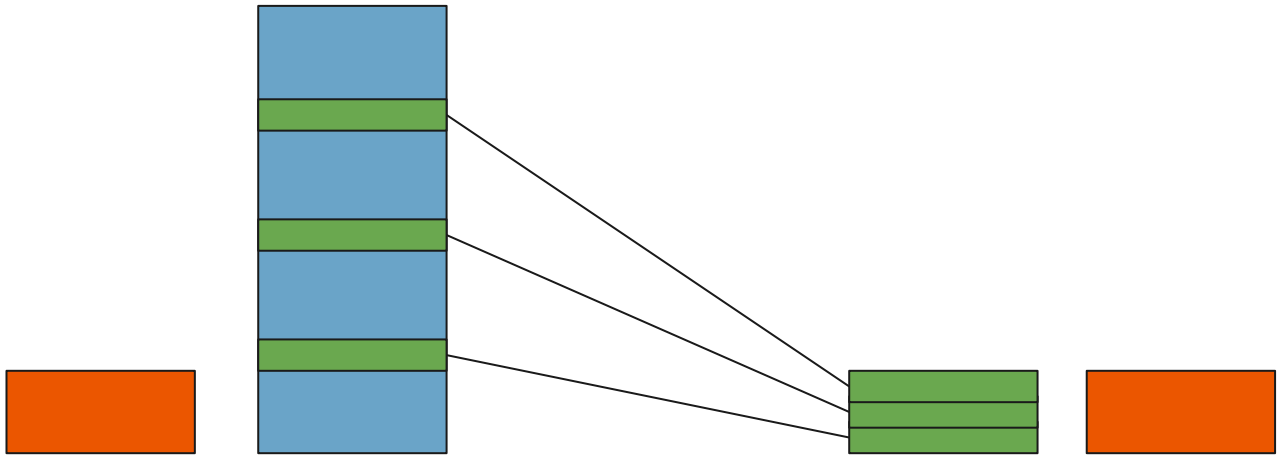
Up-Sampling : Pros and Cons

- **Pros:**
  - This method doesn't lead to information loss.
  - Performs well and gives good accuracy.

- **Cons:**
  - The random oversampling may increase the likelihood of **overfitting** occurring, since it makes exact copies of the minority class examples.
  - Increases Learning Time.

Down-sampling

Downsampling or **Undersampling** refers to remove or reduce the majority of class samples to balance the class labels. Below is downsampling Illustration.



Original dataset

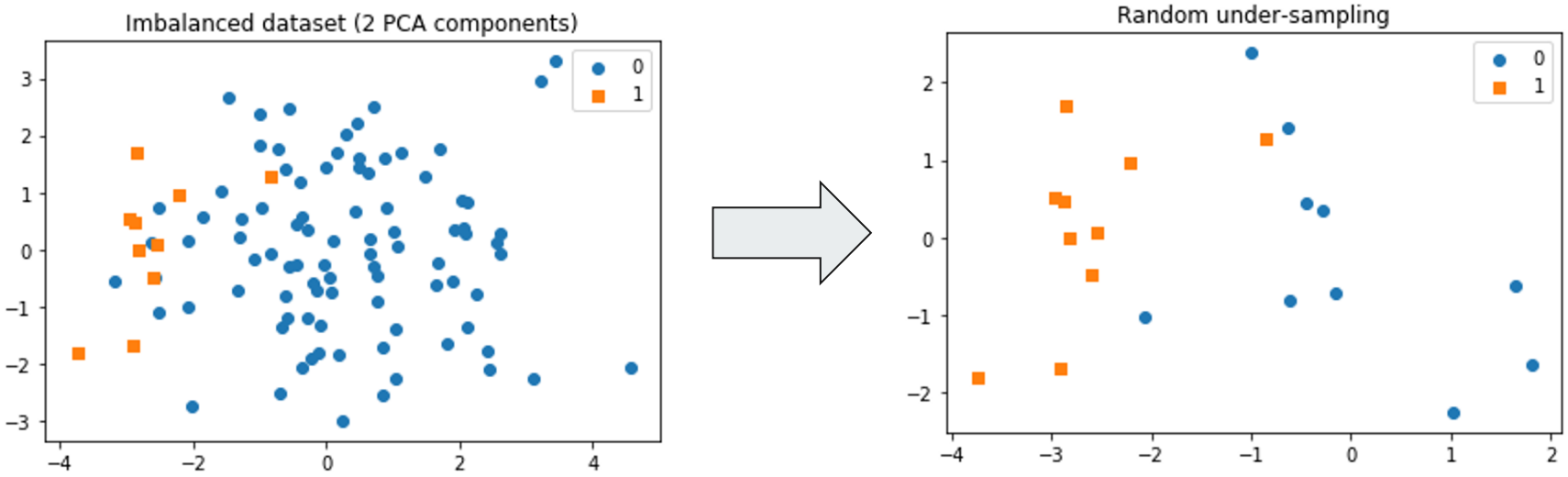
Down-sampling techniques

1. Random Under Sample
2. Tomek Link
3. ENN (Edited Nearest Neighbours)
4. NearMiss Sampling

Random Under Sample

Randomly delete examples in the majority class.

Random undersampling leads to potential loss of information - since a lot of data instances are just 'thrown away'.



Reference: https://www.kaggle.com/code/rafjaa/resampling-strategies-for-imbalanced-datasets/notebook?fbclid=IwAR271yoaSePCjNons5qWmyFcYWsL8Ecp6pqyhqYMs5njP1CgH-KS5nVGgZU

Random Under Sample : Pros and Cons

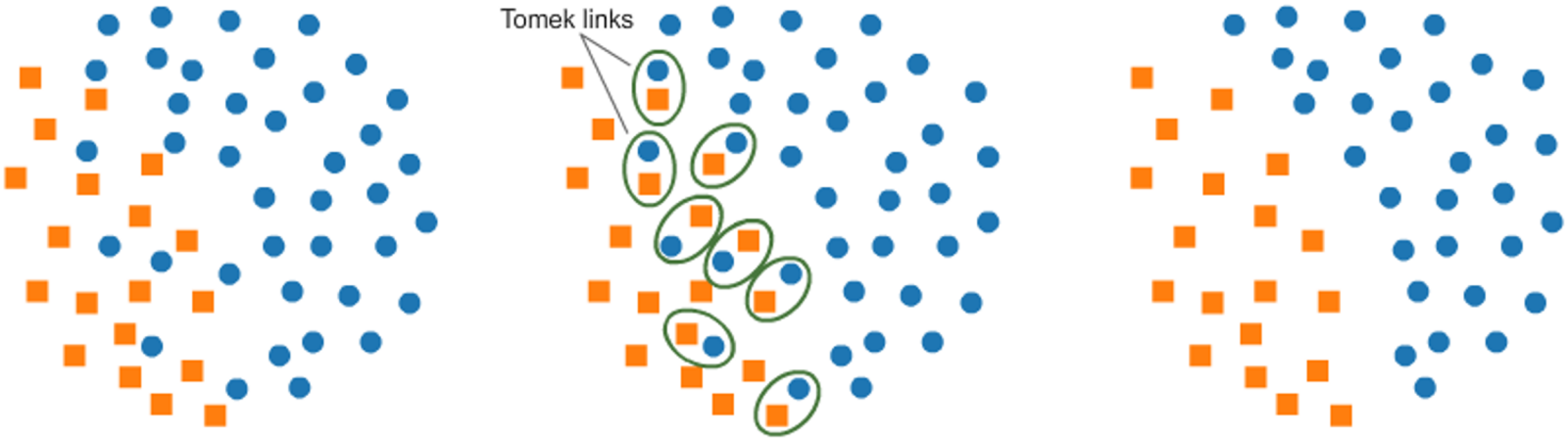
- **Pros:**
  - This method is easy to implement and also fast to execute, which is good for very large and complex datasets.
  - Less storage requirements and better run times for analyses.

- **Cons:**
  - Loss of potentially important data is particularly true with random undersampling when events are removed without any consideration for what they are and how useful they might be to the analysis.
  - To fix the problem above, methods that select examples from the majority class to delete, including the popular **Tomek Link** and, **Edited Nearest Neighbors**, and **NearMiss** will be introduced on following content.

藍色比較多，跟少量的人太像了，把藍色丟掉
現在deep learning更希望越像的東西電腦可以越區分開來

Tomek Link: Introduction

Tomek links are pairs of very close instances, but of opposite classes. The idea of Tomek links is removing the **ambiguous points** on boundary, those points are considered as **noise** which should be removed.



Paper: Two Modifications of CNN. *IEEE Transactions on Systems, Man, and Cybernetics* (1976).

Steps to find a Tomek Link

1. For an instance pair **(Xi, Yj)**,
   where Xi is an instance of the minority class; Yj is an instance of majority class.

1. Compute **z=0.5(Xi+Yj)**

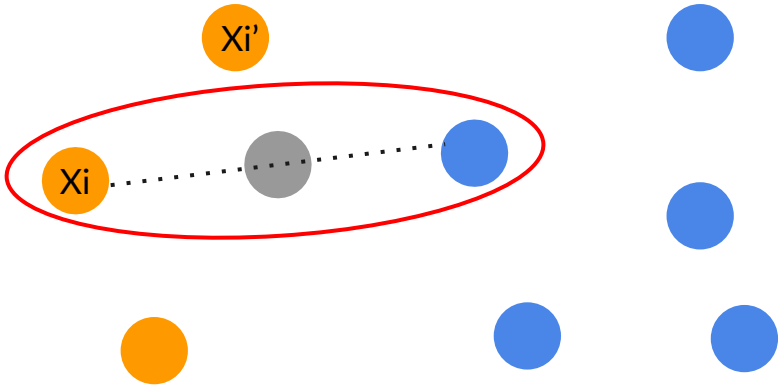
1. If there is a **Xi'** **or** **Yj'** such that:
   d(**Xi'**, **z**) ≤ d(**Xi**, **z**) **or** d(**Yj'**, **z**) ≤ d(**Yj**, **z**)
   Then,(Xi, Yj) is a **Tomek link**.

- Xi' is an instance of the majority class but Xi'≠Xi.
- Yj' is an instance of the minority class but Yj'≠Yj.
- d() means the distance between two instances.

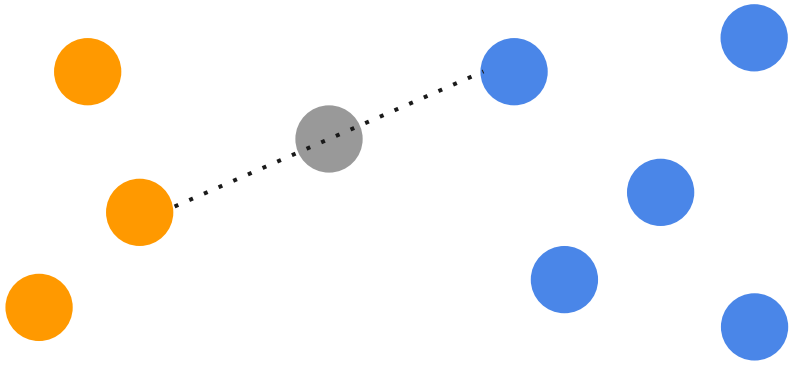
Tomek Link: Example

Practically, we remove the instance from the **majority class** in a Tomek Link.
In this case, it is the blue one.

There is an ambiguous point.



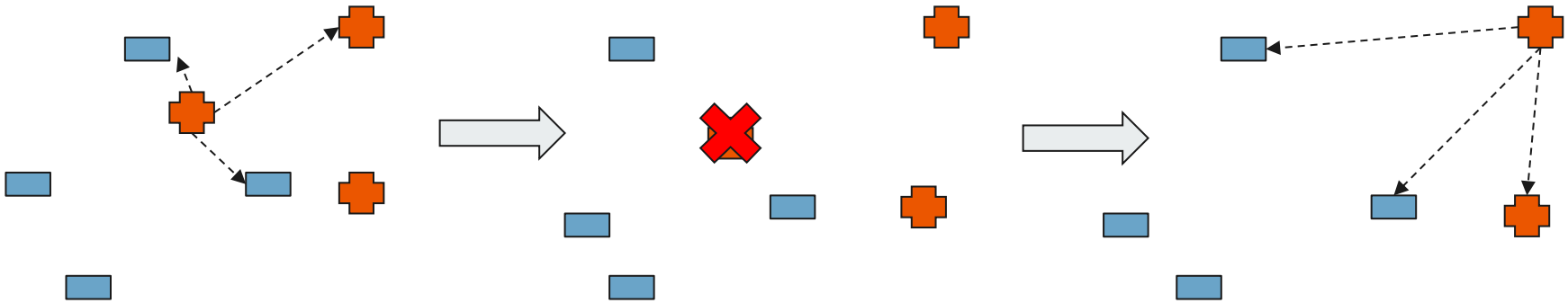
There is no ambiguous point.



ENN (Edited Nearest Neighbours)

This technique removes any instance from the majority class whose class label is different from the class label of **more than half** of its nearest neighbors.

Like idea of Tomek links to remove some ambiguous points ,but **ENN** using the **K Nearest Neighbours** method to delete the majority class which are surounding by minority class.



For example,we choose 3 nearest neighbors.

Remove this majority sample.

Move on next majority sample to repeat the above steps.

NearMiss Sampling

- NearMiss is based on k-nearest neighbors.
- NearMiss tries to make data points (including majorities and minorities) more evenly distributed by removing distant majorities.
- NearMiss is suitble for the task where an enormous gap of sample number exists between the majority and minority classes.

NearMiss Sampling

Here are the different versions of this algorithm:

- **NearMiss-1**
- **NearMiss-2**
- **NearMiss-3**

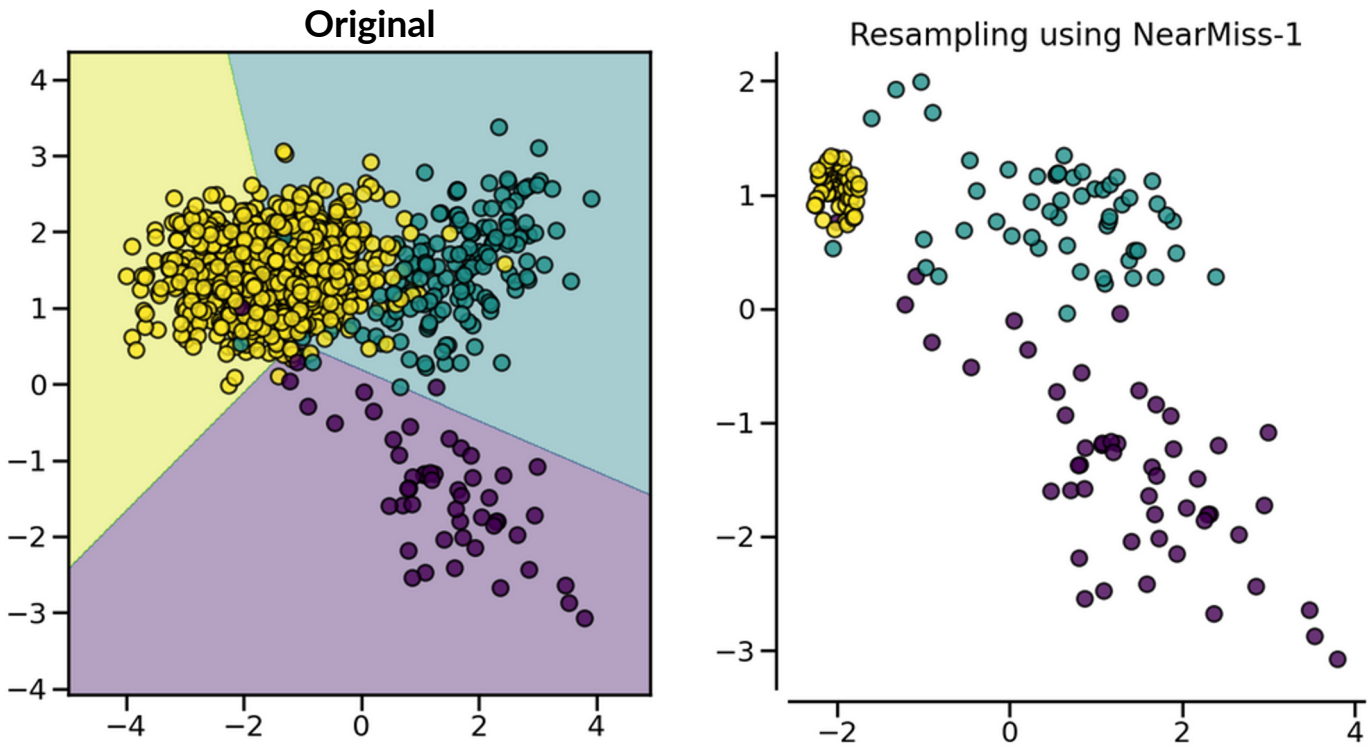
We introduce these three versions in detail in following content.

NearMiss-1 and NearMiss-2

Steps	NearMiss-1	NearMiss-2
1	Compute the avg. distance **D** between every sample of majority class and it's **K nearest** samples of minority.	Compute the avg. distance **D** between every sample of majority class and it's **K farthest** samples of minority.
2	Choose **N** samples of majority class whose avg. distances are smallest to keep.	

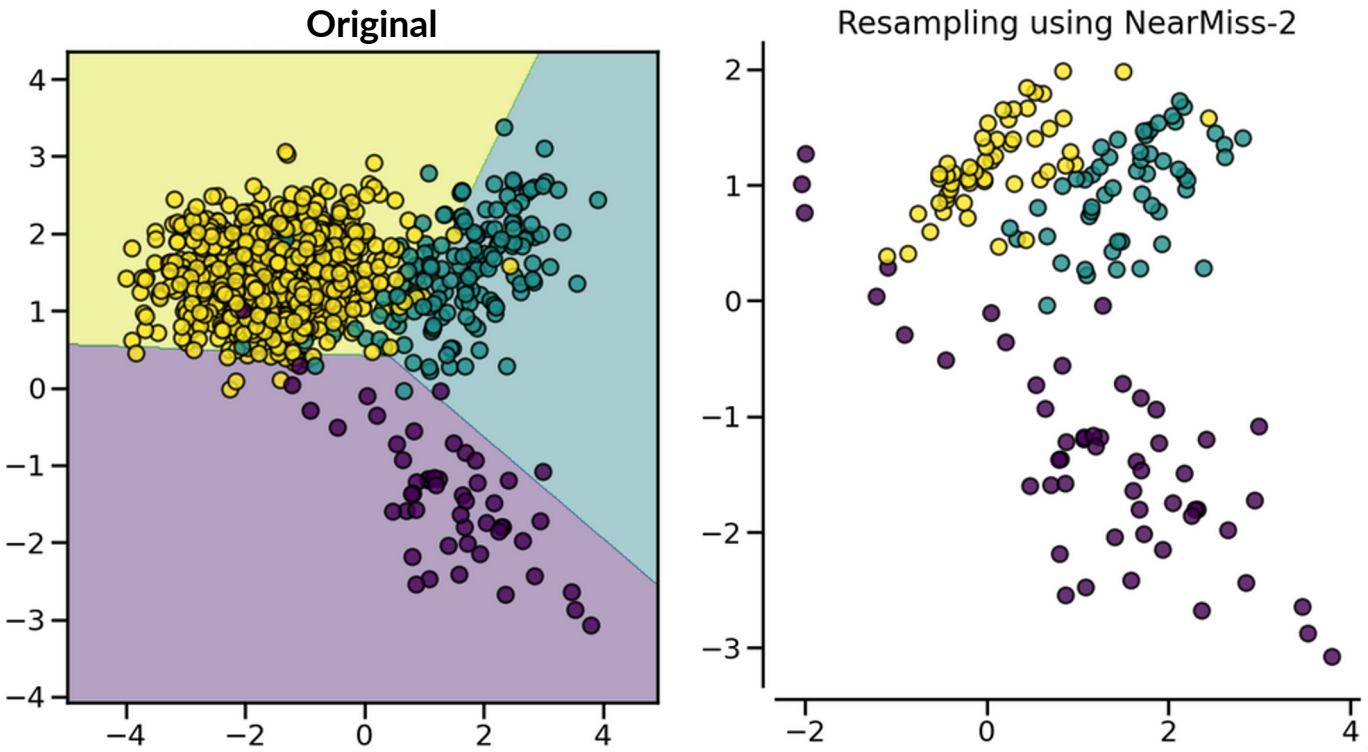
- NearMiss-1 shows a simple and direct method to keep the samples of majority class with K-nearst neighbors.
- Samples of majority class **may not be evenly distributed around minority examples**.

NearMiss-1



Source: https://imbalanced-learn.org/dev/auto_examples/under-sampling/plot_comparison_under_sampling.html#sphx-glr-auto-examples-under-sampling-plot-comparison-under-sampling-py

NearMiss-2



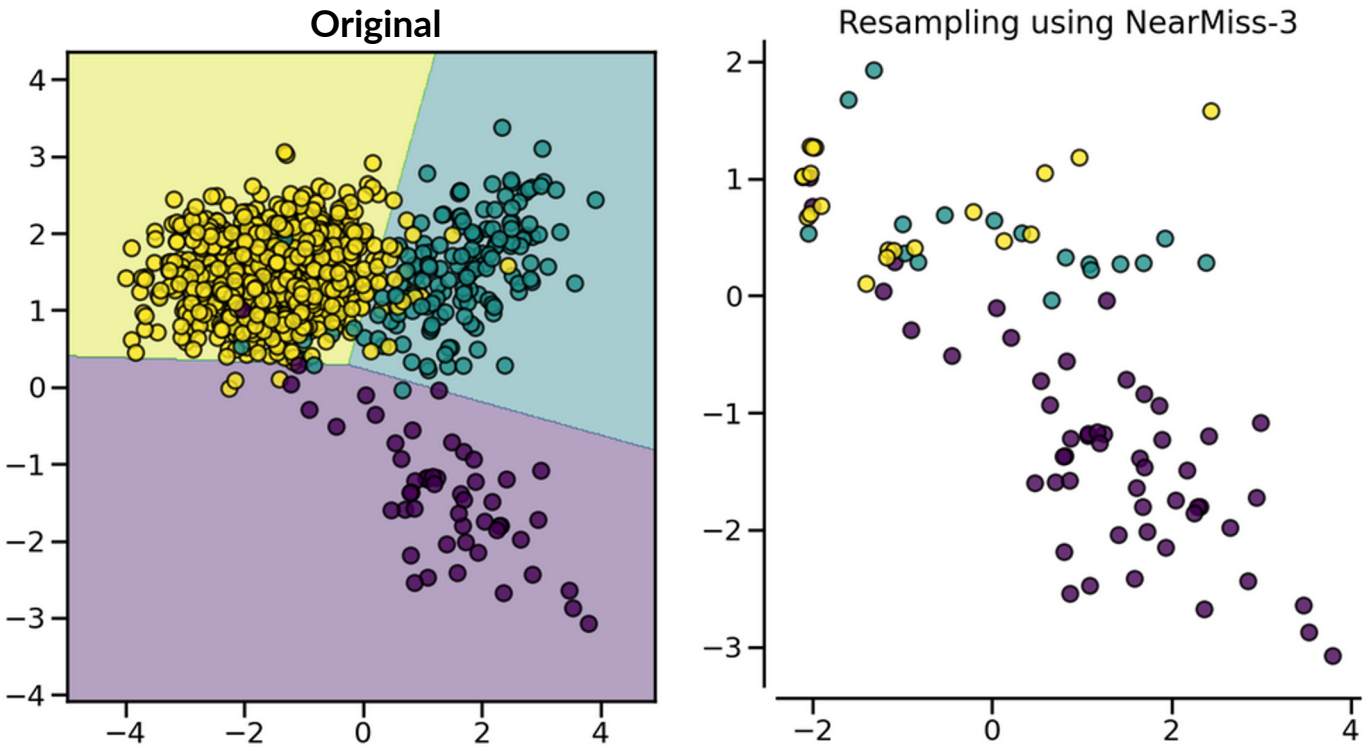
Source: https://imbalanced-learn.org/dev/auto_examples/under-sampling/plot_comparison_under_sampling.html#sphx-glr-auto-examples-under-sampling-plot-comparison-under-sampling-py

NearMiss-3

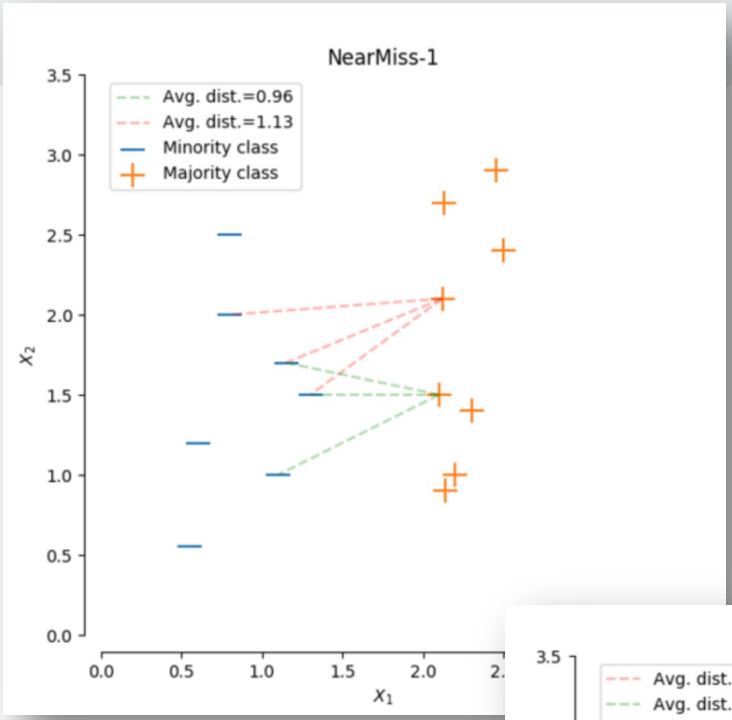
Steps	NearMiss-3
1	For each minority sample, computing the distance to all majority samples. Keep the **M** majority samples with the shortest distance.
2	For the remaining majority samples, computing average distances to **K** closest minority class.
3	Finally, keep the majority samples with the **farthest** avg. distances.

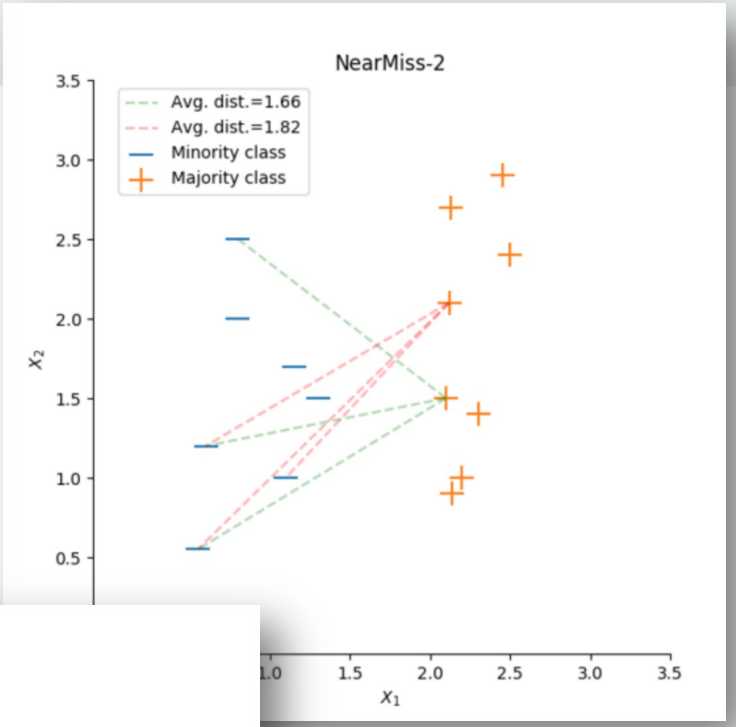
- More evenly distributed: **NearMiss-3 > NearMiss-2 > NearMiss-1**
- Can be viewed as a combination of NearMiss-1 and NearMiss-2.

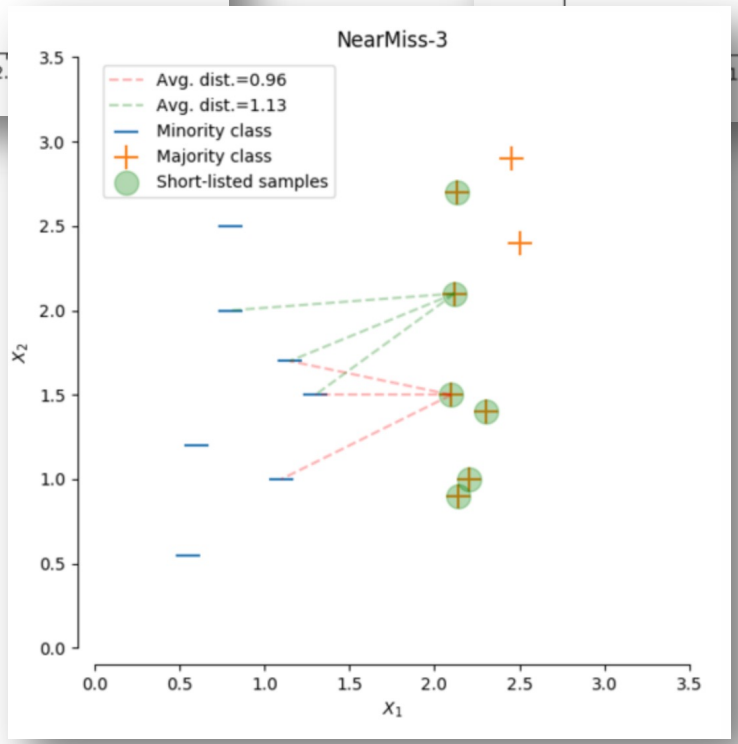
NearMiss-3



Source: https://imbalanced-learn.org/dev/auto_examples/under-sampling/plot_comparison_under_sampling.html#sphx-glr-auto-examples-under-sampling-plot-comparison-under-sampling-py


NearMiss-1
Avg. dist.=0.96
Avg. dist.=1.13
Minority class
Majority class
X_1
X_2


NearMiss-2
Avg. dist.=1.66
Avg. dist.=1.82
Minority class
Majority class
X_1
X_2


NearMiss-3
Avg. dist.=0.96
Avg. dist.=1.13
Minority class
Majority class
Short-listed samples
X_1
X_2

Cost-Sensitive Learning

- **Cost**: The penalty associated with an incorrect prediction.
- The goal of cost-sensitive learning:
  - Give uneven penalties to the majority class and the minority class.
  - **Minimize the cost** of a model **with the given weighted penalties**.

Traditional classification

- Take binary cross-entropy as an example.

$$\mathcal{L} = -\frac{1}{n}\sum_{1}^{n}[\underbrace{y_i \log(\hat{y}_i)}_{\text{for class: 1}} + \underbrace{(1-y_i)\log(1-\hat{y}_i)}_{\text{for class: 0}}]$$

Classification with differnt penalties

- Take binary cross-entropy as an example.

$$\mathcal{L} = -\frac{1}{n}\sum_{1}^{n}[w_1\, y_i \log(\hat{y}_i) + w_0\,(1-y_i)\log(1-\hat{y}_i)]$$

for class: 1 for class: 0

An example with Cost-Sensitive Learning

Actual	Predict	Penalty
0	0	0
1	0	**88**
1	1	0
0	1	**5**
0	0	0
0	0	0

0: Covid-19 negatives
1: Covid-19 positives

- To minimize the total cost, a machine learning algorithm should pay more attention to the class with a higher penalty.

Cost-Sensitive Learning : Pros and Cons

- **Pros:**
  - Don’t need to remove or copy any examples on the dataset.Therefore,you can reduce the disadvantages of sampling methods.

- **Cons:**
  - There are not cost-sensitive implementations of all learning algorithms and therefore an approach using sampling is the only option.

Paper : Cost-Sensitive Learning vs. Sampling: Which is Best for Handling Unbalanced Classes with Unequal Error Costs?

Conclusion

- **Undersampling** is a useful way to balance imbalanced datasets. But it is imperative for data scientists to consider the **information potentially lost** through undersampling.

- Consider the various sampling techniques as well as **combination** undersampling and oversampling methods. You can experiment with these methods to determine the most effective way to resample their data and obtain the most accurate results. Combination of over- and under-sampling methods