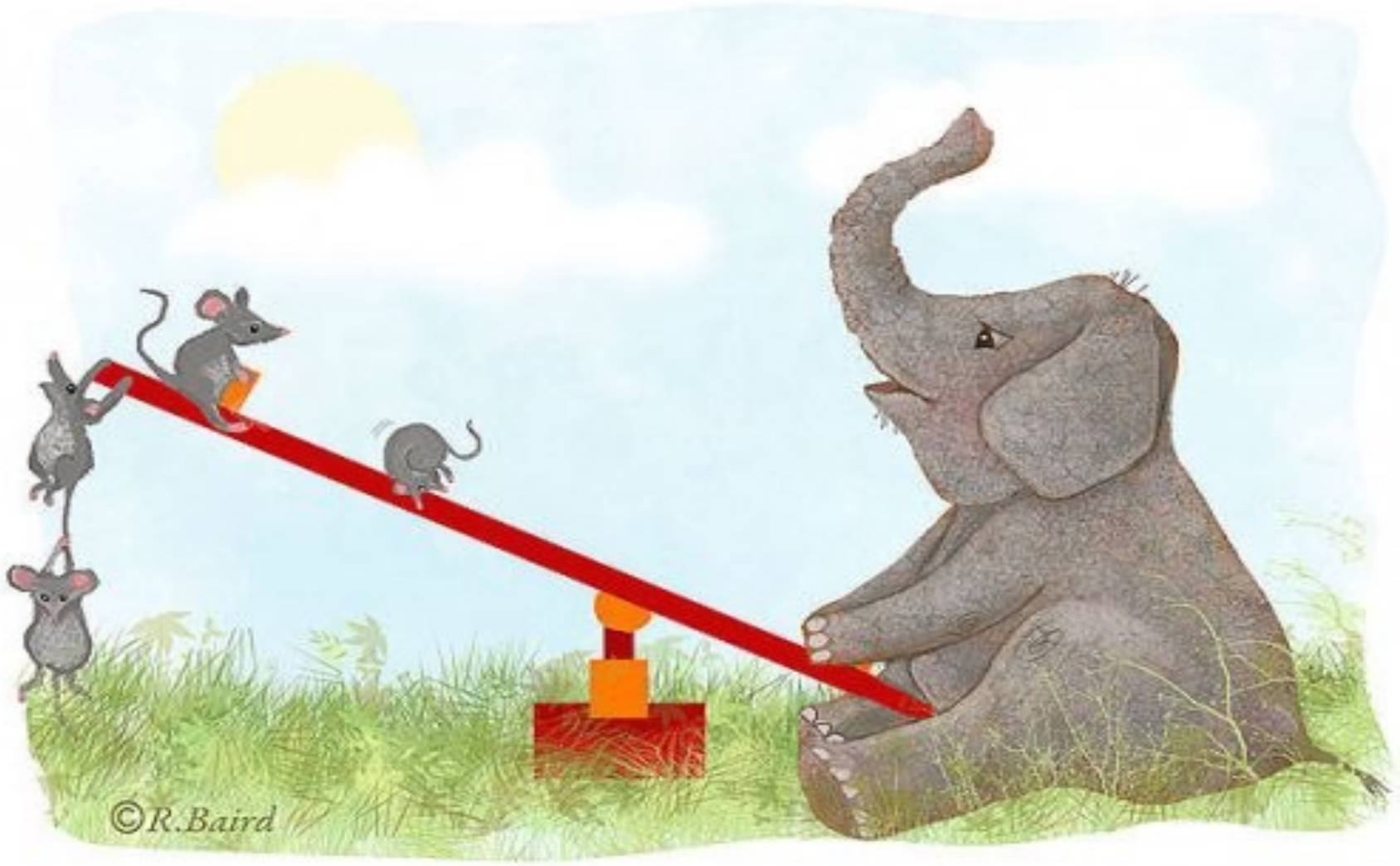
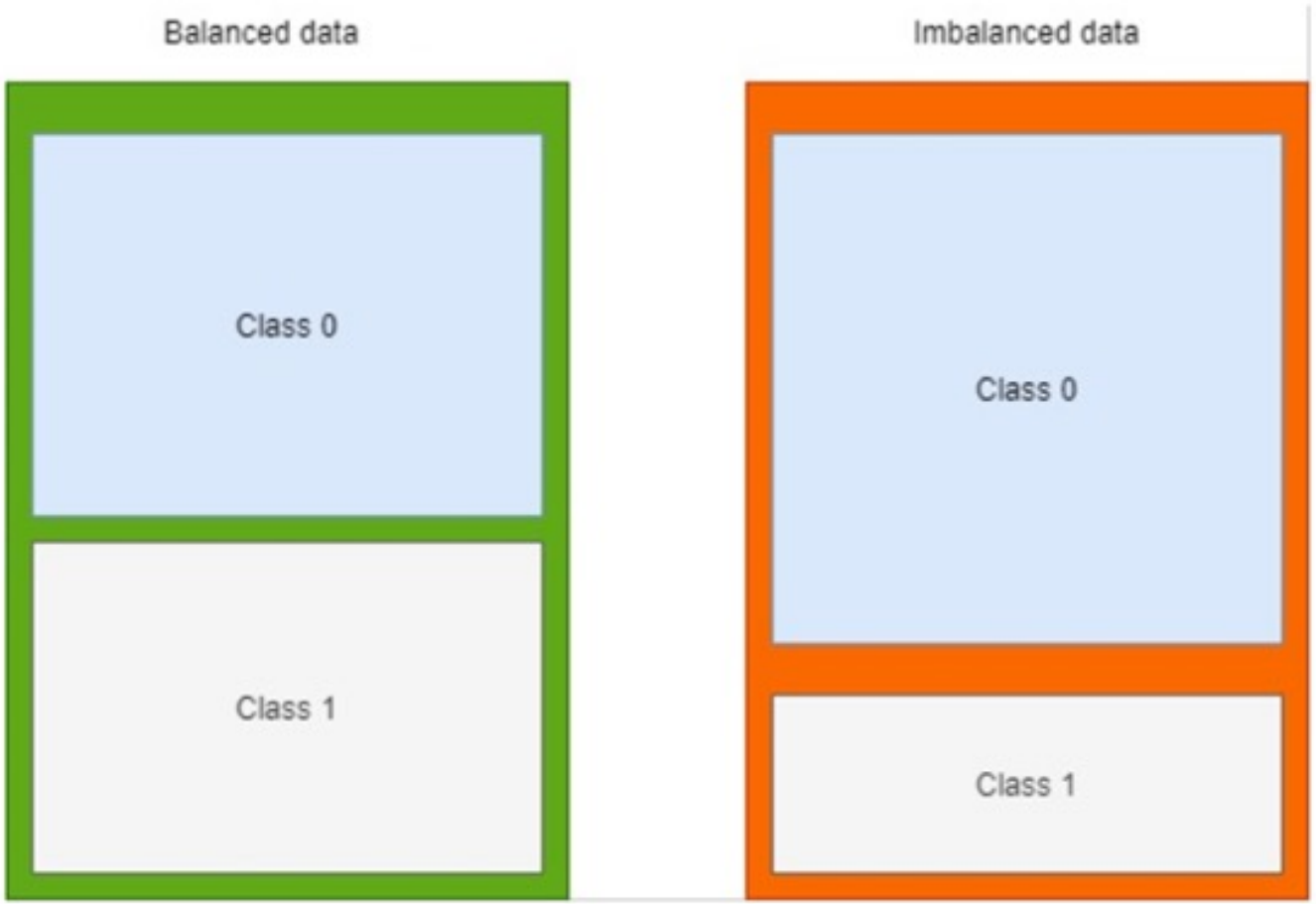



# Handle Imbalance Dataset Problem

Mì Ai

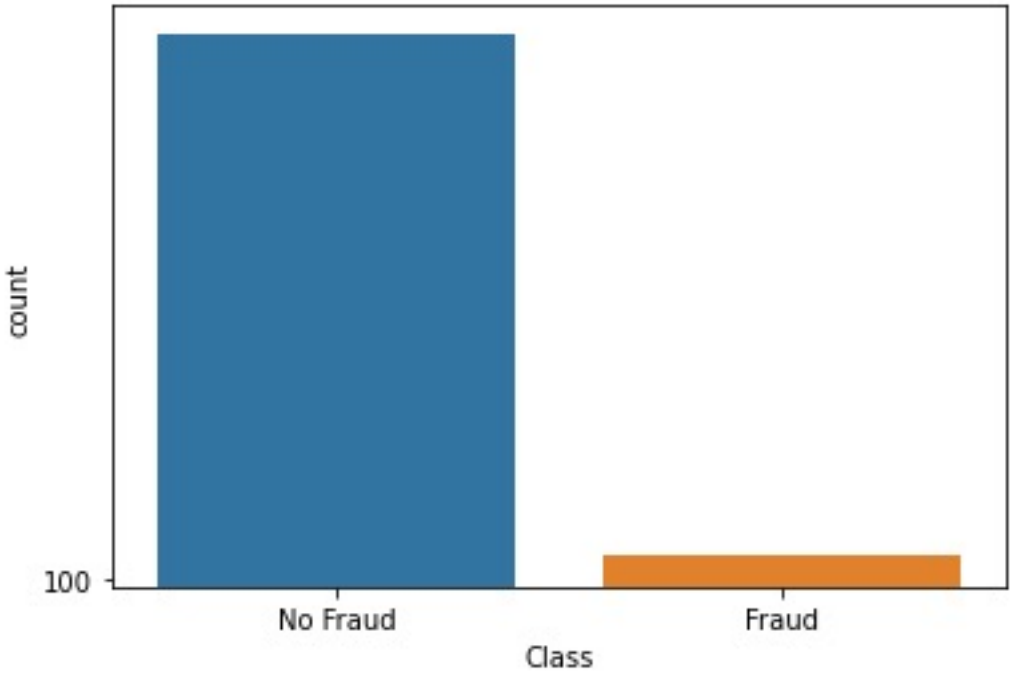
# Vấn đề Imbalance Data

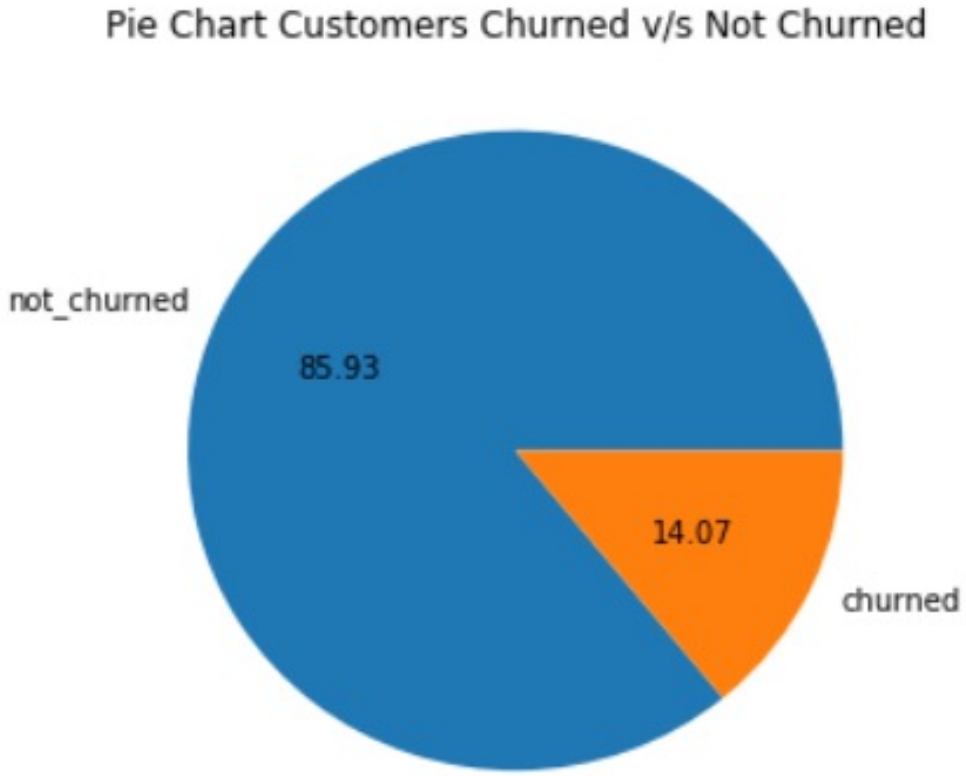



Majority

Minority

# Vấn đề Imbalance Data




What’s happen if above dataset is balance?

# Tác hại của Imbalance

- Hầu hết các thuật toán Phân lớp trong ML hoạt động tốt với balance dataset.
- Khi dataset không balance:
  - Model có thiên hướng predict ra lớp Majority để tăng Accuracy.
  - Accuracy không còn tác dụng đánh giá.

# Tác hại của Imbalance

Ví dụ:

- Ta có 100 điểm dữ liệu bệnh nhân cần dự đoán Ung thư
- Trong đó:
  - 99 điểm là không bị Ung thư
  - 1 điểm là bị ung thư
- Nếu ta sử dụng một model luôn trả về Không bị ung thư -> model sẽ đạt accuracy 99% (Quá tuyệt!!!!???).
- Khi ta train model thì model sẽ có xu hướng trả về Không bị ung thư nhằm đạt Accuracy cao.

# Các phương pháp xử lý

- Thay đổi metric đánh giá model
- Undersampling
- Oversampling
- Class weighted
- Ensemble & Boosting

# Các phương pháp xử lý

- Thay đổi metric đánh giá model

```
from sklearn import metrics
import numpy as np
y_pred = np.around(model.predict(x_test))
print(metrics.classification_report(y_test, y_pred))
```

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.92 | 0.93 | 0.92 | 12500 |
| 1 | 0.93 | 0.92 | 0.92 | 12500 |
| micro avg | 0.92 | 0.92 | 0.92 | 25000 |
| macro avg | 0.92 | 0.92 | 0.92 | 25000 |
| weighted avg | 0.92 | 0.92 | 0.92 | 25000 |

# Các phương pháp xử lý

- Thay đổi metric đánh giá model

## “Oánh giá” model AI theo cách Mì ăn liền – Chương 2. Precision, Recall và F Score

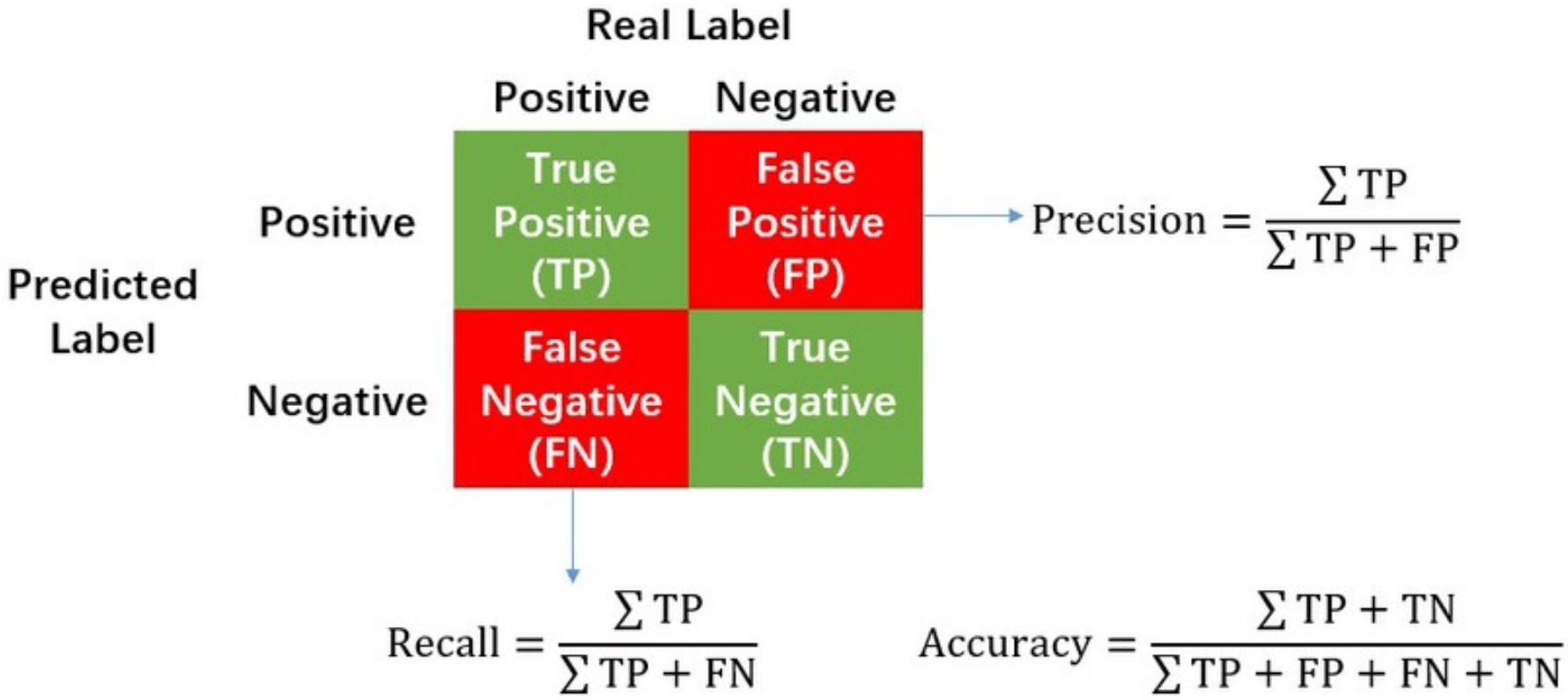
Nguyễn Chiến Thắng 16/06/2020 Basic 2 Comments

Lượt xem: 10,814

Hello tuần mới anh em Mì AI, như vậy trong bài trước mình đã cùng nhau khởi động series về oánh giá model AI với hai khái niệm Loss và Accuracy **tại đây**. Hôm nay chúng ta sẽ đi tiếp series này với việc tìm hiểu các khái niệm Precision, Recall va F Score nhé.

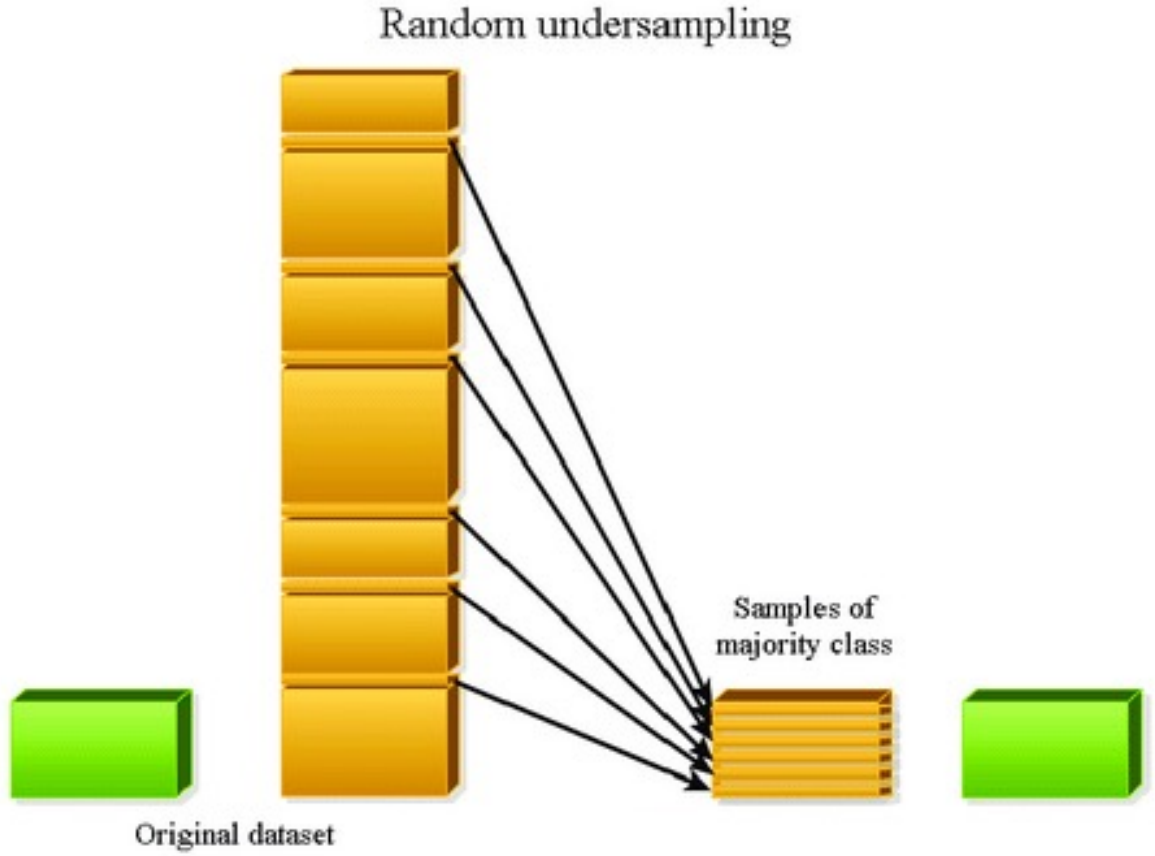
# Các phương pháp xử lý

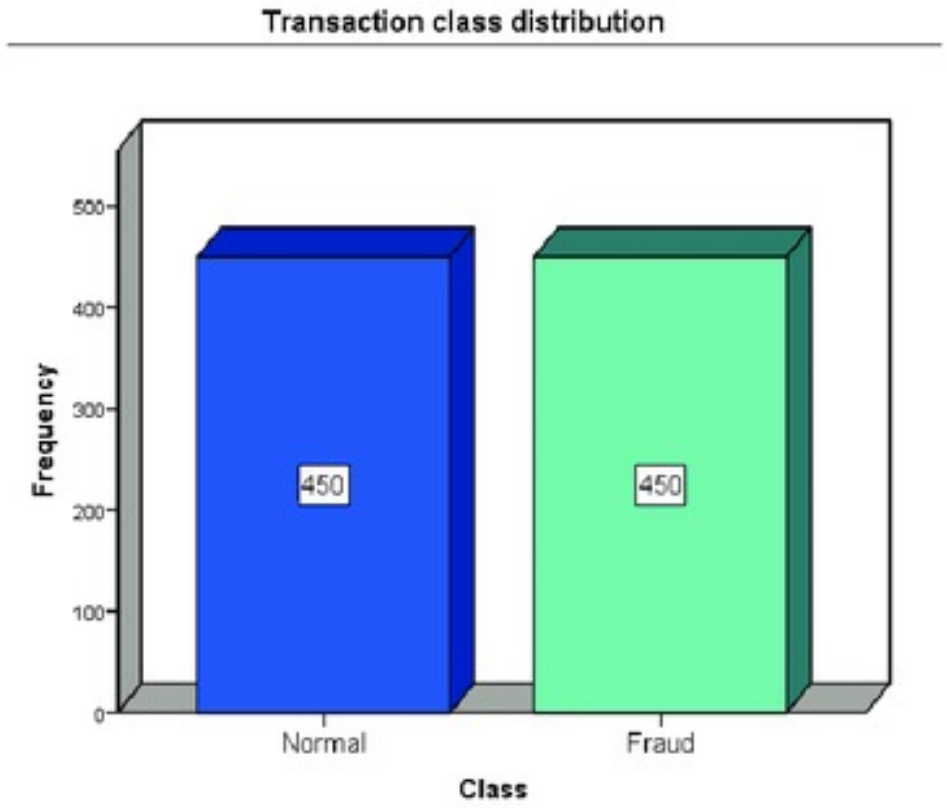
- Thay đổi metric đánh giá model



# Undersampling

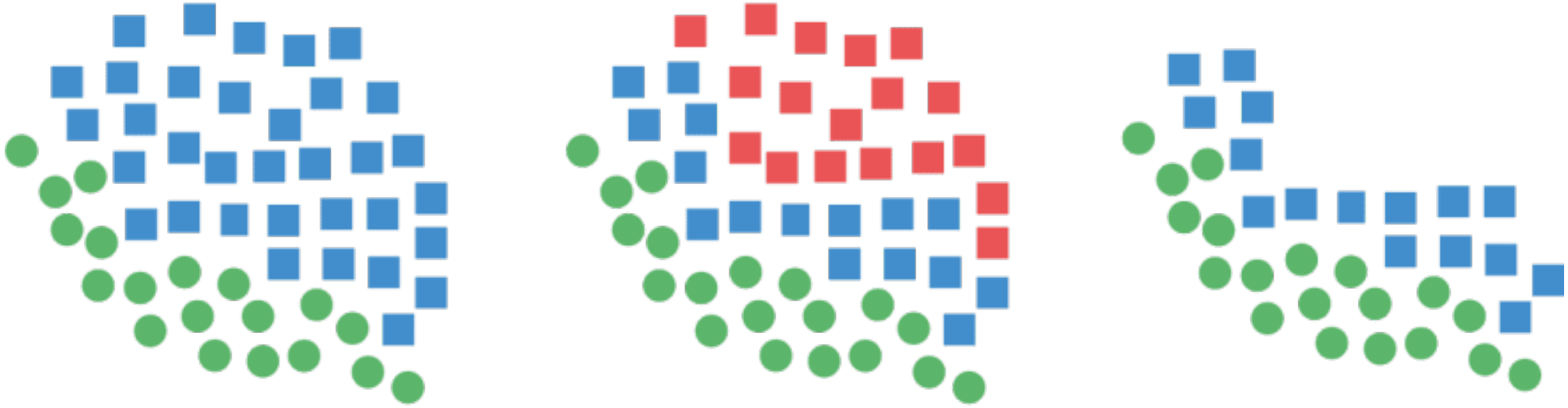
- Random Undersampling



# Undersampling

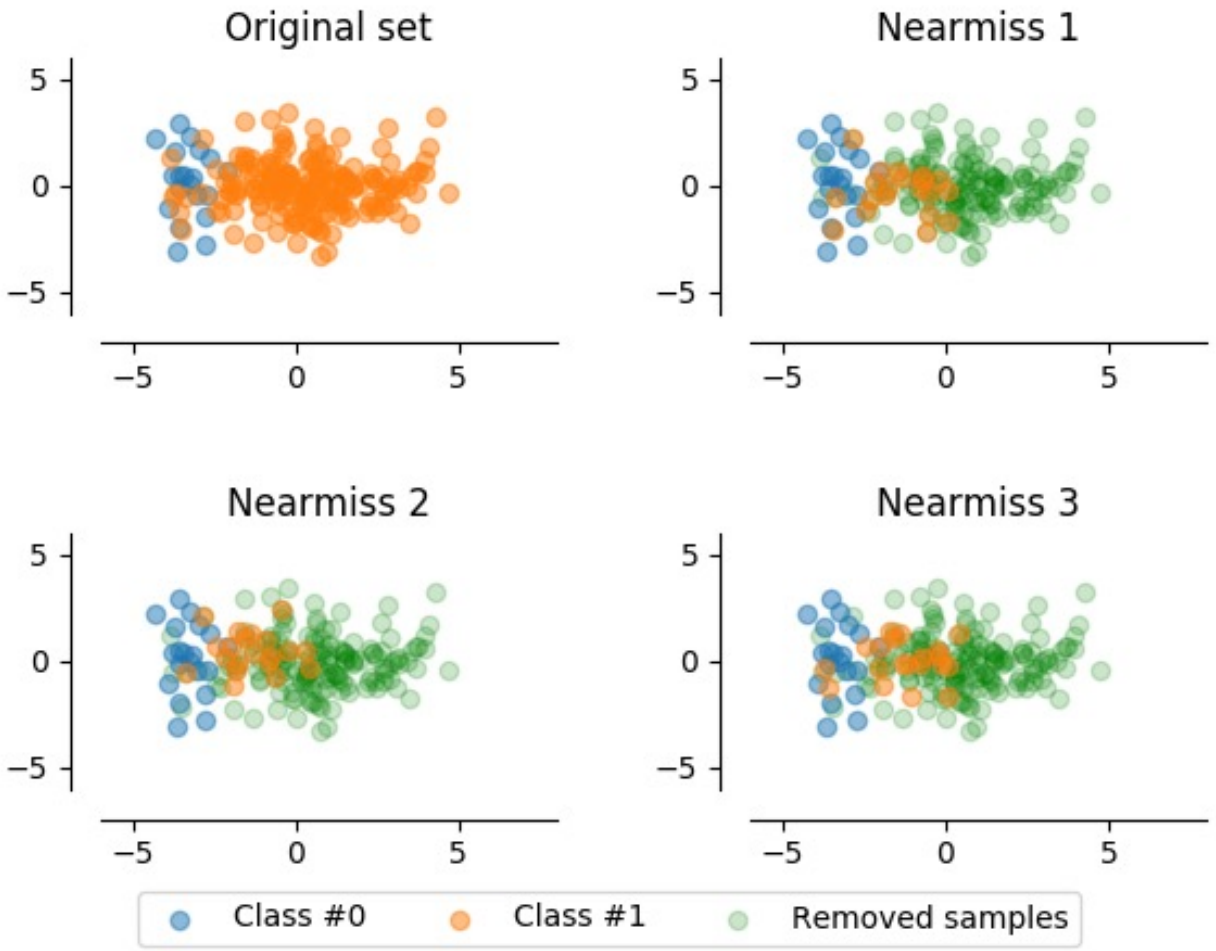
- NearMiss



# Undersampling

- **NearMiss-1**: Majority class examples with minimum average distance to three closest minority class examples.
- **NearMiss-2**: Majority class examples with minimum average distance to three furthest minority class examples.
- **NearMiss-3**: Majority class examples with minimum distance to each minority class example.

# Undersampling

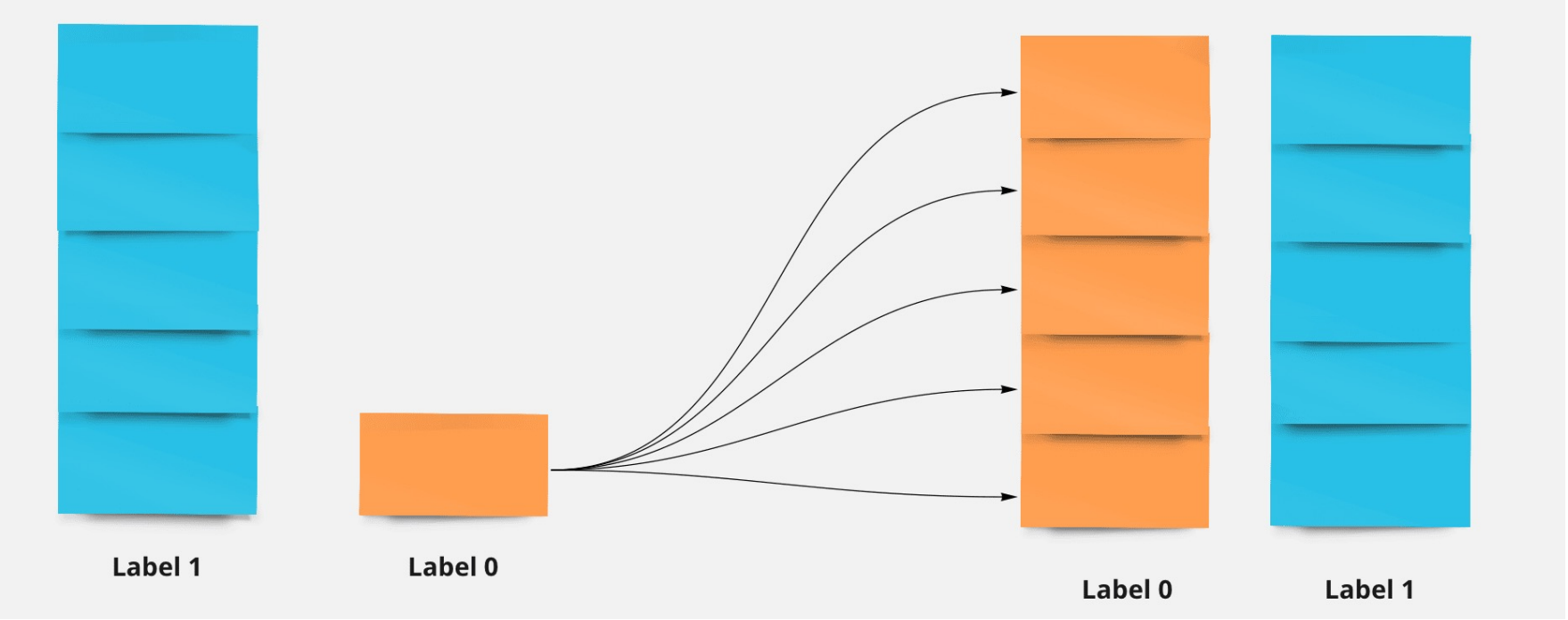


# Oversampling

- Random Oversampling
- Cluster Oversampling
- SMOTE

# Oversampling

- Random Oversampling



# Oversampling

- Cluster-Based Over Sampling

In this case, the K-means clustering algorithm is independently applied to minority and majority class instances. This is to identify clusters in the dataset. Subsequently, each cluster is oversampled such that all clusters of the same class have an equal number of instances and all classes have the same size.

# Oversampling

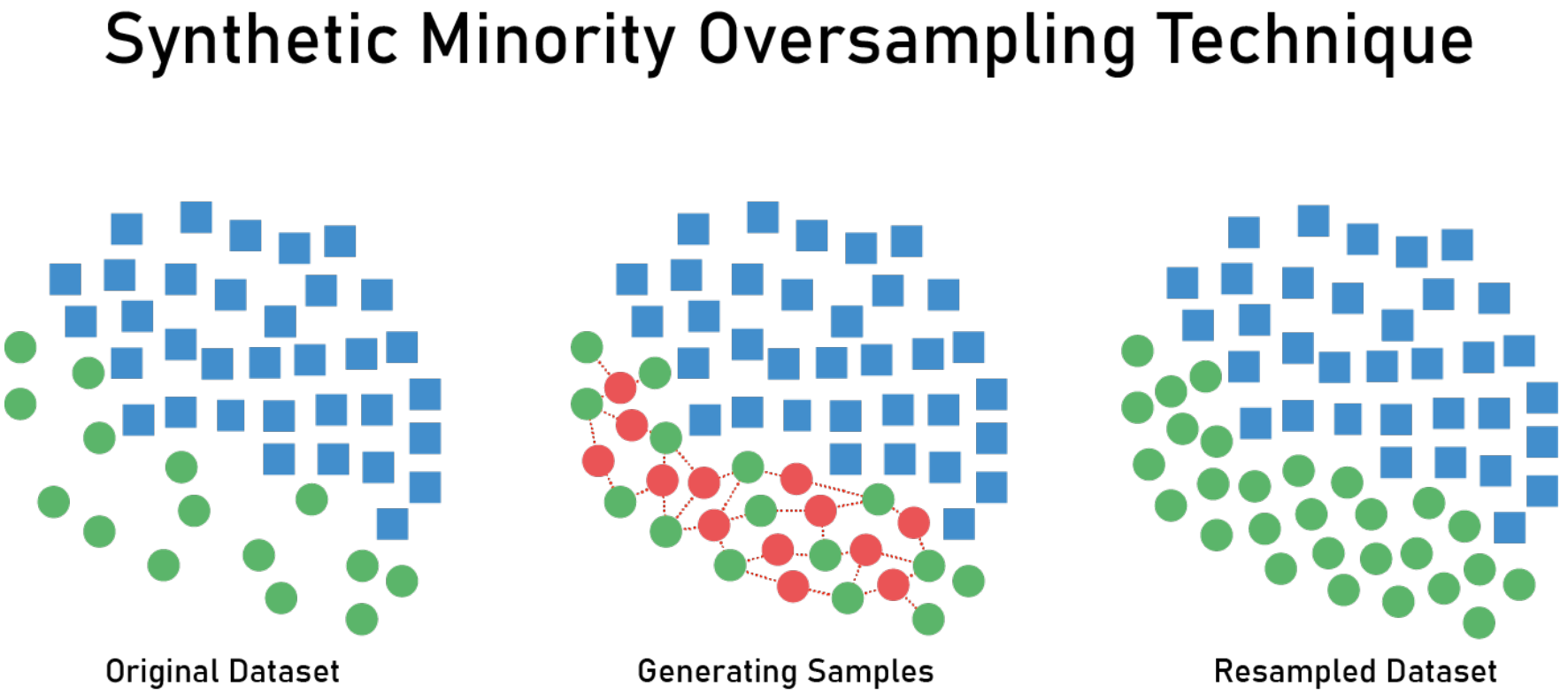
- Cluster-Based Over Sampling

- **Majority Class Clusters**
    1. Cluster 1: 150 Observations
    2. Cluster 2: 120 Observations
    3. Cluster 3: 230 observations
    4. Cluster 4: 200 observations
    5. Cluster 5: 150 observations
    6. Cluster 6: 130 observations
- **Minority Class Clusters**
    1. Cluster 1: 8 Observations
    2. Cluster 2: 12 Observations

- **Majority Class Clusters**
    1. Cluster 1: 170 Observations
    2. Cluster 2: 170 Observations
    3. Cluster 3: 170 observations
    4. Cluster 4: 170 observations
    5. Cluster 5: 170 observations
    6. Cluster 6: 170 observations
- **Minority Class Clusters**
    1. Cluster 1: 250 Observations
    2. Cluster 2: 250 Observations

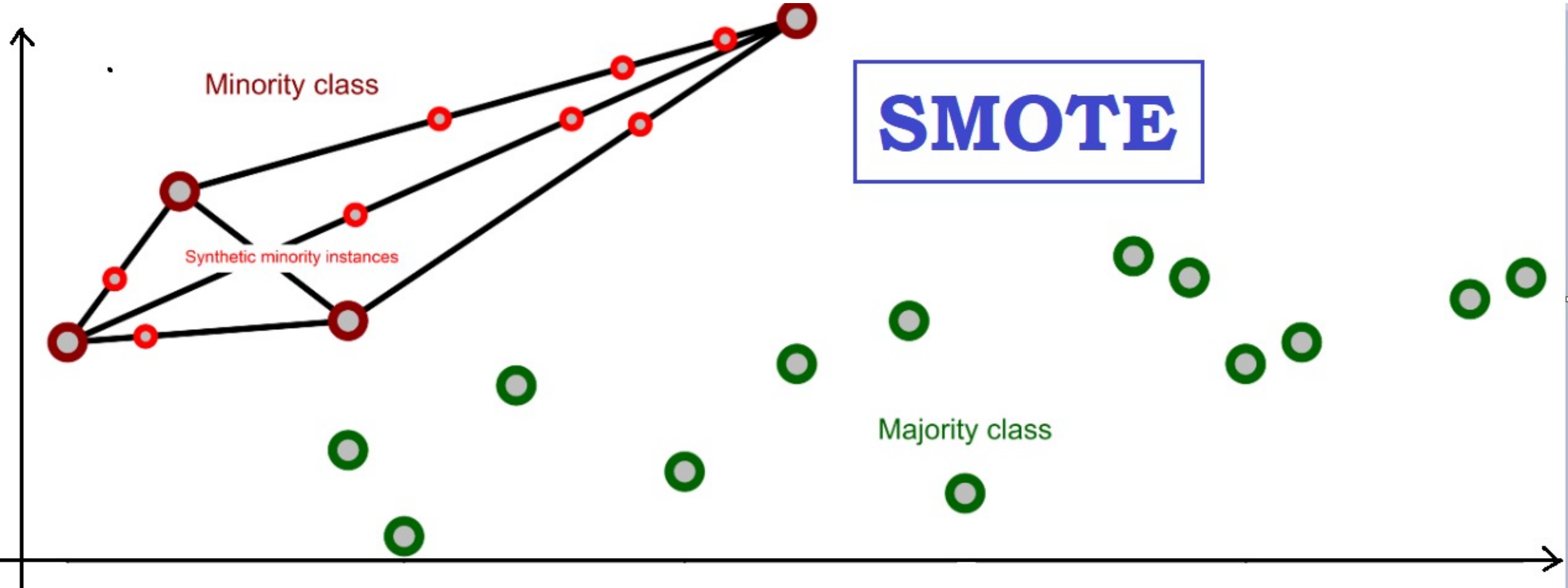
# Oversampling

- SMOTE



# Oversampling

- SMOTE



# Oversampling

- SMOTE

- **Advantages**
  - Mitigates the problem of overfitting caused by random oversampling as synthetic examples are generated rather than replication of instances
  - No loss of useful information
- **Disadvantages**
  - While generating synthetic examples SMOTE does not take into consideration neighboring examples from other classes. This can result in increase in overlapping of classes and can introduce additional noise
  - SMOTE is not very effective for high dimensional data

MSMOTE is an improve method

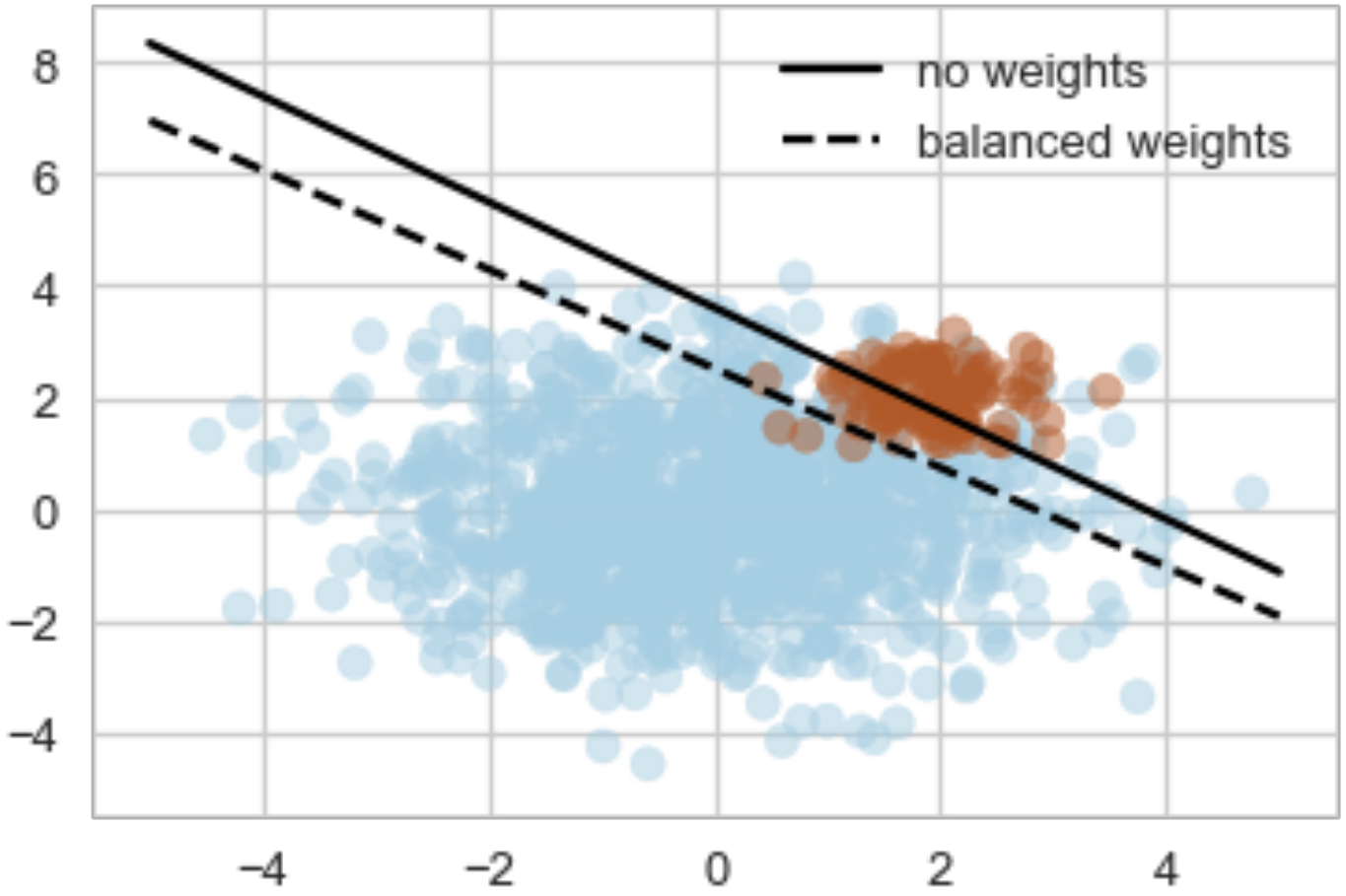
# Class_weights

```
wj=n_samples / (n_classes * n_samplesj)
```

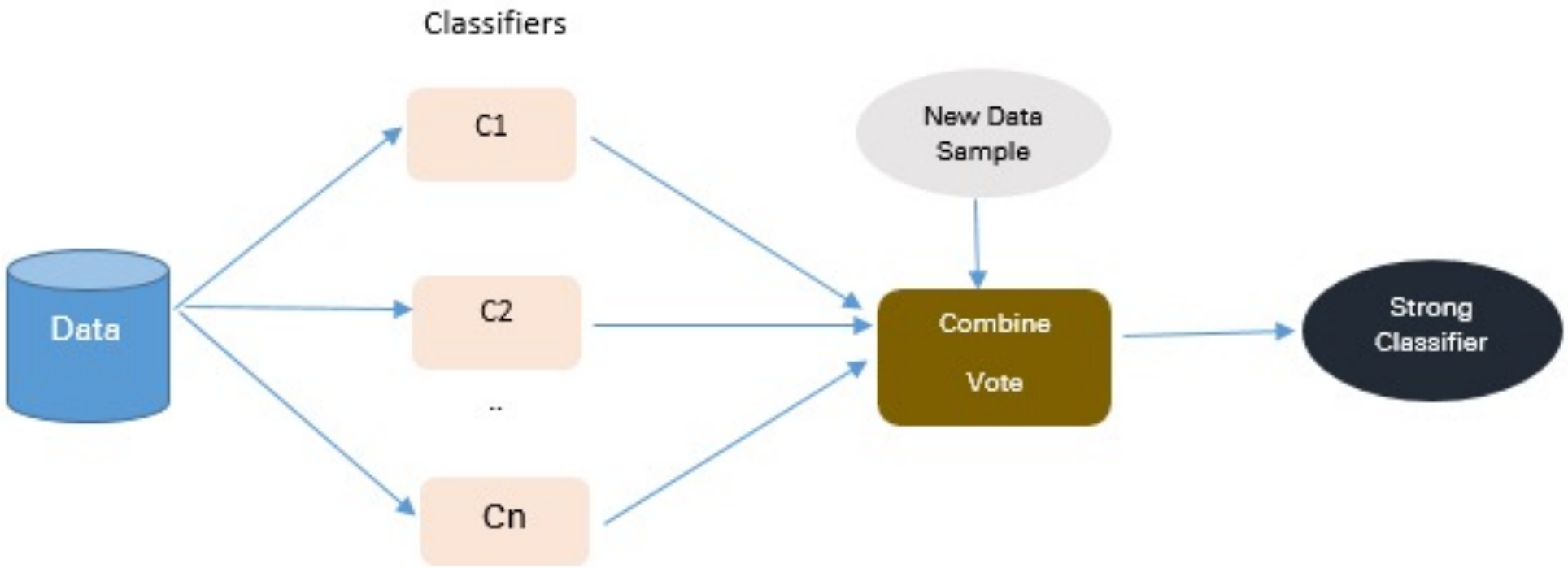
ad locum,

- WJ is the weight of each class (J is the class)
- n_ Samples is the total number of samples or rows in the dataset
- n_ Classes is the total number of unique classes in the target
- n_ Samplesj is the total number of rows of the corresponding class

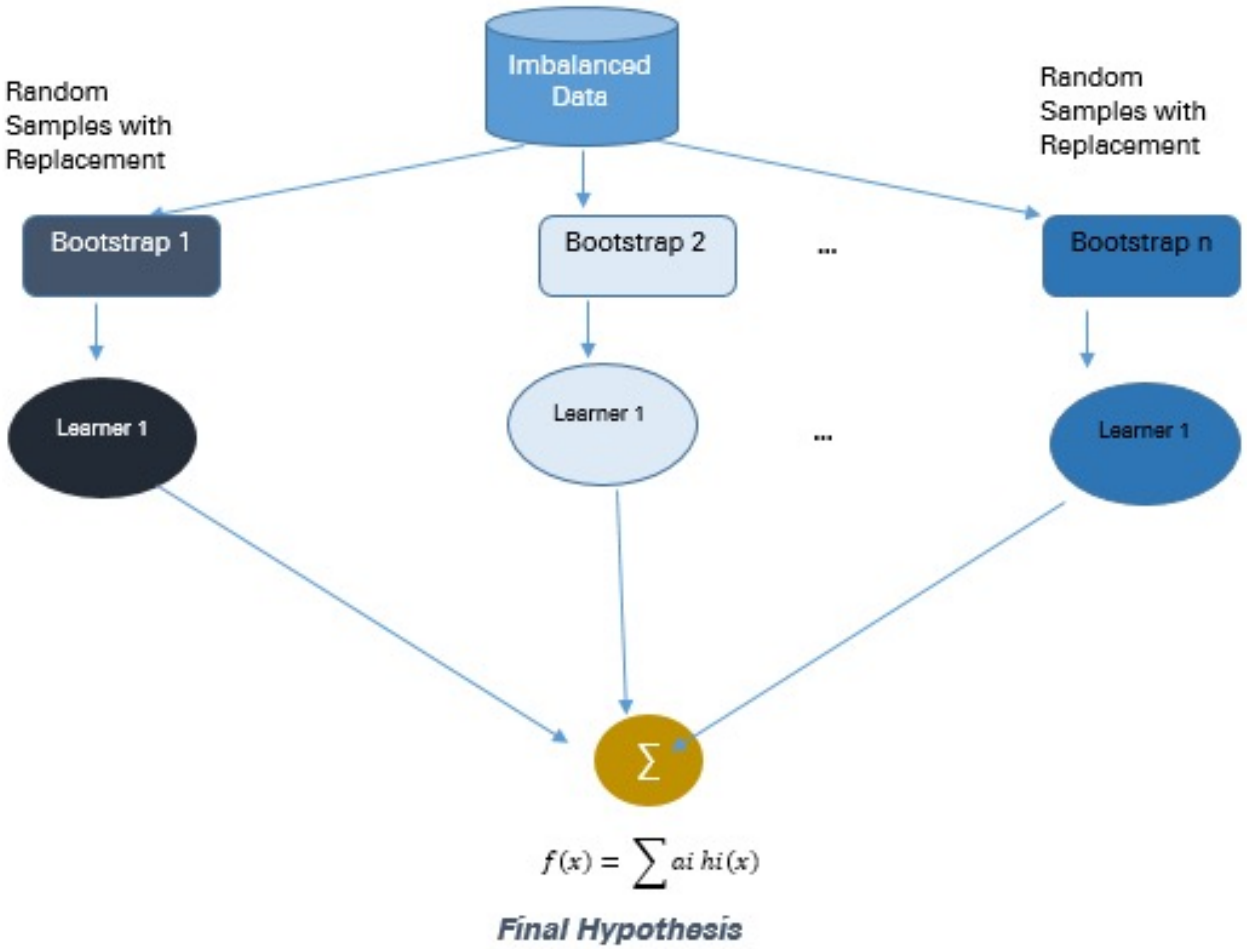
# Class_weights



# Ensemble & Boosting



# Ensemble & Boosting



# Ensemble & Boosting

The final tactic we'll consider is using tree-based algorithms. Decision trees often perform well on imbalanced datasets because their hierarchical structure allows them to learn signals from both classes.

In modern applied machine learning, tree ensembles (Random Forests, Gradient Boosted Trees, etc.)

We will try:

- RandomForest
- GradientBoosting
- XGBoost

# Handson