# RUSBoost : A Hybrid Approach to Alleviating Class Imbalance

19. 05. 17
Yongwon Jo
Data Mining & Quality Analytics Lab.

DMQA

# Contents



DMQA

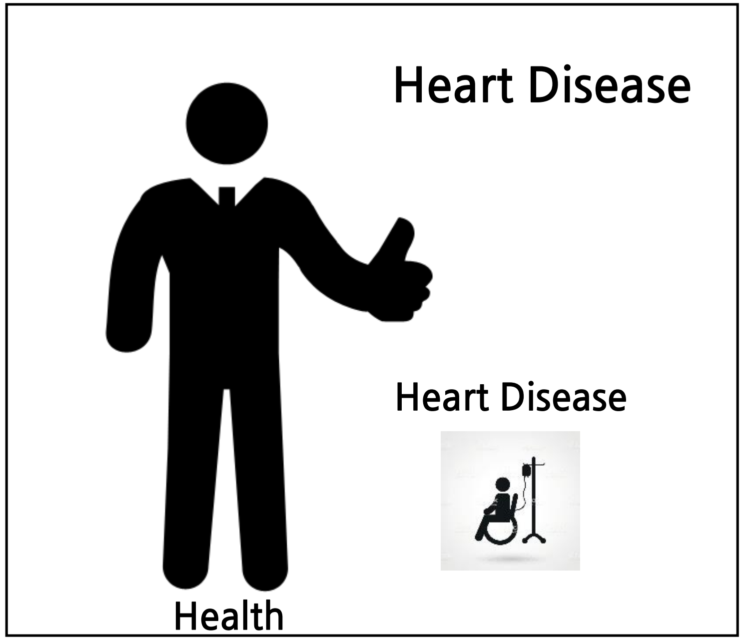
# Contents

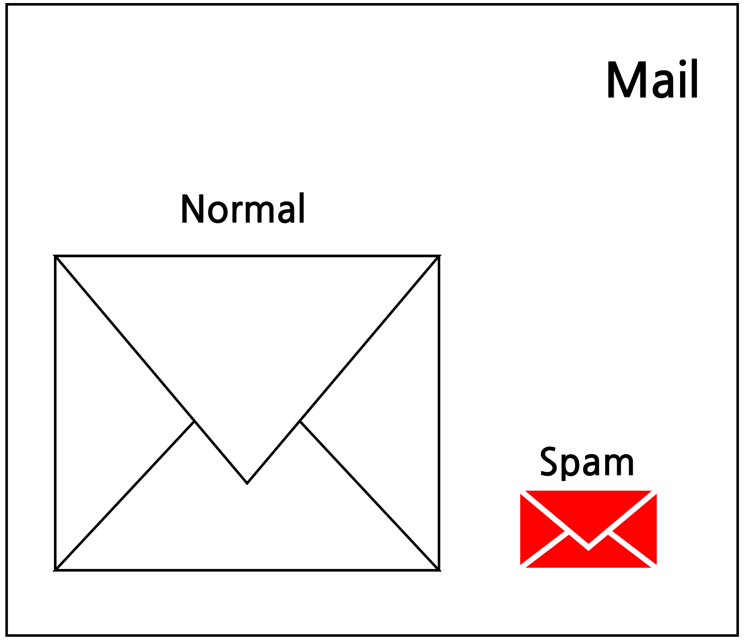
# I. Introduction to Class Imbalance problem

## ❖ Class Imbalance problem

- It is the problem in classification where **the total number of a class of data (positive) is far less** than the total number of another class of data (negative).

- **This problem exists for many domains.**



# I. Introduction to Class Imbalance problem

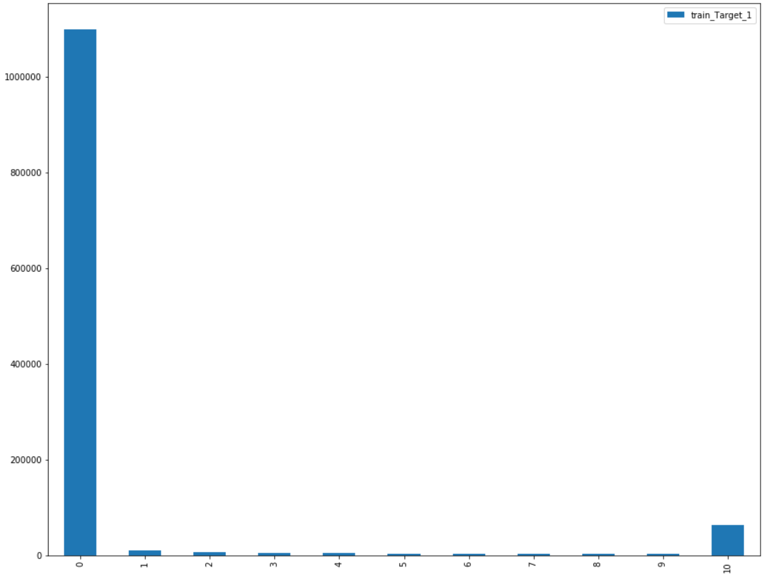
## ❖ Class Imbalance problem

- Below plots are the class imbalance situation I actually saw.

# I. Introduction to Class Imbalance problem

## ❖ Class Imbalance problem

- Below plots are the class imbalance situation I actually saw.
- It is a bar that **shows the output quantity divided by the remain quantity.**



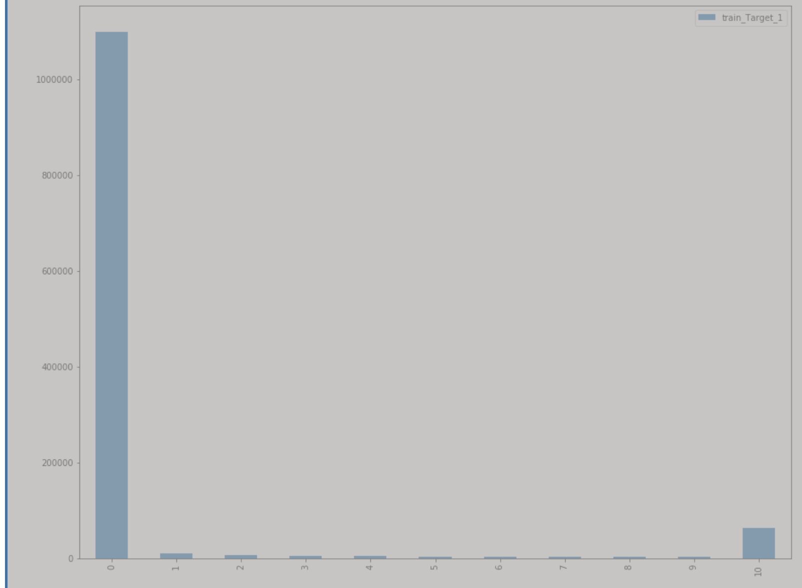

| | Count |
|---|---|
| 0 | 1099648 |
| 1 | 10595 |
| 2 | 7091 |
| 3 | 5347 |
| 4 | 4738 |
| 5 | 3998 |
| 6 | 3333 |
| 7 | 3312 |
| 8 | 2947 |
| 9 | 2950 |
| 10 | 64432 |

# I. Introduction to Class Imbalance problem

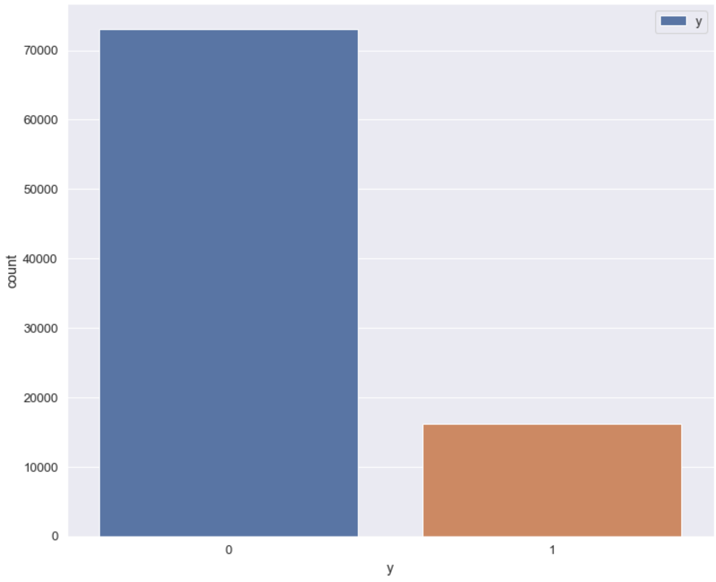
## ❖ Class Imbalance problem

- Below plots are the class imbalance situation I actually saw.

- It is a bar chart about **whether the lot will be put into the next process.**



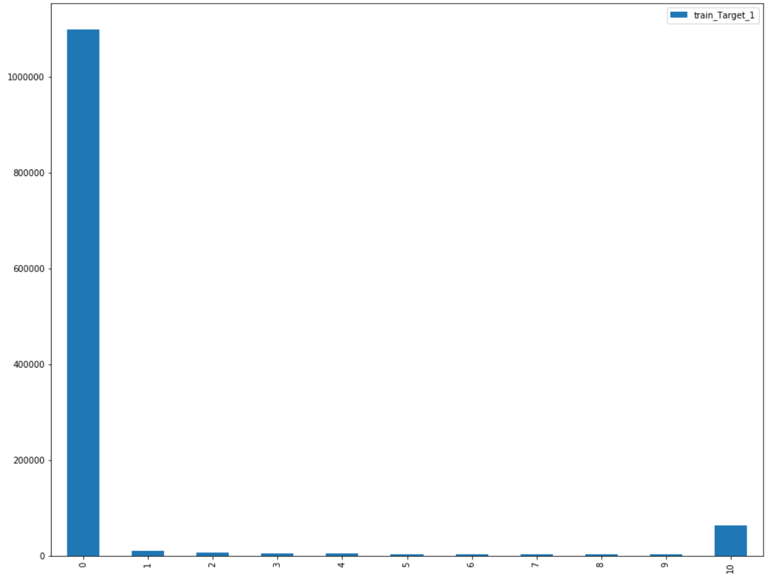

| | Count |
|---|---|
| 0 | 1099648 |
| 1 | 10595 |
| 2 | 7091 |
| 3 | 5347 |
| 4 | 4738 |
| 5 | 3998 |
| 6 | 3333 |
| 7 | 3312 |
| 8 | 2947 |
| 9 | 2950 |
| 10 | 64432 |



# I. Introduction to Class Imbalance problem

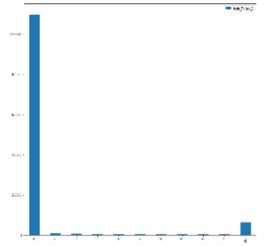
## ❖ Class Imbalance problem

- RandomForestClassifier(max_depth=30, n_estimators=200)
- Train dataset -> Accuracy : 0.90089 | F1 : 0.652276 | Recall : 0.98723 | Precision : 0.45484
- **Validation dataset -> Accuracy : 0.83854 | F1 : 0.19458 | Recall : 0.29088 | Precision : 0.14618**




| | Count |
|---|---|
| 0 | 1099648 |
| 1 | 10595 |
| 2 | 7091 |
| 3 | 5347 |
| 4 | 4738 |
| 5 | 3998 |
| 6 | 3333 |
| 7 | 3312 |
| 8 | 2947 |
| 9 | 2950 |
| 10 | 64432 |

# I. Introduction to Class Imbalance problem



| | Count |
|---|---|
| 0 | 1099648 |
| 1 | 10595 |
| 2 | 7091 |
| 3 | 5347 |
| 4 | 4738 |
| 5 | 3998 |
| 6 | 3333 |
| 7 | 3312 |
| 8 | 2947 |
| 9 | 2950 |
| 10 | 64432 |

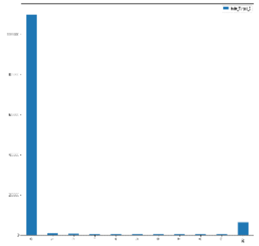
## ❖ Class Imbalance problem

- RandomForestClassifier(max_depth=30, n_estimators=200)
- Train dataset -> Accuracy : 0.90089 | F1 : 0.652276 | Recall : 0.98723 | Precision : 0.45484
- **Validation dataset -> Accuracy : 0.83854 | F1 : 0.19458 | Recall : 0.29088 | Precision : 0.14618**

| | | Predicted | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Actual | 0 | **515126** | 86 | 1 | 6 | 8 | 7 | 3 | 2 | 4 | 2 | 8572 |
| | 1 | 6900 | **528** | 17 | 2 | 0 | 0 | 2 | 0 | 0 | 2 | 1585 |
| | 2 | 5375 | 147 | **20** | 8 | 3 | 0 | 0 | 0 | 1 | 0 | 1596 |
| | 3 | 4154 | 50 | 11 | **9** | 1 | 1 | 2 | 0 | 1 | 0 | 1535 |
| | 4 | 3808 | 37 | 5 | 5 | **3** | 1 | 0 | 0 | 0 | 0 | 1436 |
| | 5 | 3407 | 27 | 1 | 2 | 1 | **0** | 1 | 0 | 1 | 1 | 1450 |
| | 6 | 2870 | 16 | 0 | 1 | 0 | 2 | **2** | 1 | 0 | 0 | 1370 |
| | 7 | 2928 | 11 | 1 | 0 | 0 | 0 | 1 | **1** | 0 | 0 | 1464 |
| | 8 | 2596 | 18 | 0 | 3 | 2 | 0 | 0 | 0 | **1** | 1 | 1383 |
| | 9 | 2742 | 7 | 0 | 0 | 0 | 2 | 1 | 0 | 1 | **0** | 1466 |
| | 10 | 56073 | 37 | 6 | 0 | 4 | 3 | 4 | 4 | 4 | 4 | **71017** |

# I. Introduction to Class Imbalance problem

## ❖ Class Imbalance problem



| | Count |
|---|---|
| 0 | 1099648 |
| 1 | 10595 |
| 2 | 7091 |
| 3 | 5347 |
| 4 | 4738 |
| 5 | 3998 |
| 6 | 3333 |
| 7 | 3312 |
| 8 | 2947 |
| 9 | 2950 |
| 10 | 64432 |

- RandomForestClassifier(max_depth=30, n_estimators=200)
- Train dataset -> Accuracy : 0.90089 | F1 : 0.652276 | Recall : 0.98723 | Precision : 0.45484
- **Validation dataset -> Accuracy : 0.83854 | F1 : 0.19458 | Recall : 0.29088 | Precision : 0.14618**

| | | Predicted | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **0** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | **10** |
| Actual | 0 | **515126** | 86 | 1 | 6 | 8 | 7 | 3 | 2 | 4 | 2 | 8572 |
| | 1 | 6900 | **528** | 17 | 2 | 0 | 0 | 2 | 0 | 0 | 2 | 1585 |
| | 2 | 5375 | 147 | **20** | 8 | 3 | 0 | 0 | 0 | 1 | 0 | 1596 |
| | 3 | 4154 | 50 | 11 | **9** | 1 | 1 | 2 | 0 | 1 | 0 | 1535 |
| | 4 | 3808 | 37 | 5 | 5 | **3** | 1 | 0 | 0 | 0 | 0 | 1436 |
| | 5 | 3407 | 27 | 1 | 2 | 1 | **0** | 1 | 0 | 1 | 1 | 1450 |
| | 6 | 2870 | 16 | 0 | 1 | 0 | 2 | **2** | 1 | 0 | 0 | 1370 |
| | 7 | 2928 | 11 | 1 | 0 | 0 | 0 | 1 | **1** | 0 | 0 | 1464 |
| | 8 | 2596 | 18 | 0 | 3 | 2 | 0 | 0 | 0 | **1** | 1 | 1383 |
| | 9 | 2742 | 7 | 0 | 0 | 0 | 2 | 1 | 0 | 1 | **0** | 1466 |
| | 10 | 56073 | 37 | 6 | 0 | 4 | 3 | 4 | 4 | 4 | 4 | **71017** |

# Contents



DMQA

# II. How to solve Class Imbalance problem

## ❖ Sampling Techniques

- Over Sampling vs. Under Sampling

## ❖ Algorithm Techniques

- AdaBoost ,……

## ❖ Feature selection Techniques

- Lots of feature selection techniques

출처 : Class Imbalance Problem in Data Mining: Review
1Mr.Rushi Longadge, 2 Ms. Snehlata S. Dongre, 3Dr. Latesh Malik

# II. How to solve Class Imbalance problem

## ❖ Sampling Techniques

- Over Sampling vs. Under Sampling

## ❖ Algorithm Techniques

- Boosting, AdaBoost,

## ❖ Feature selection Techniques

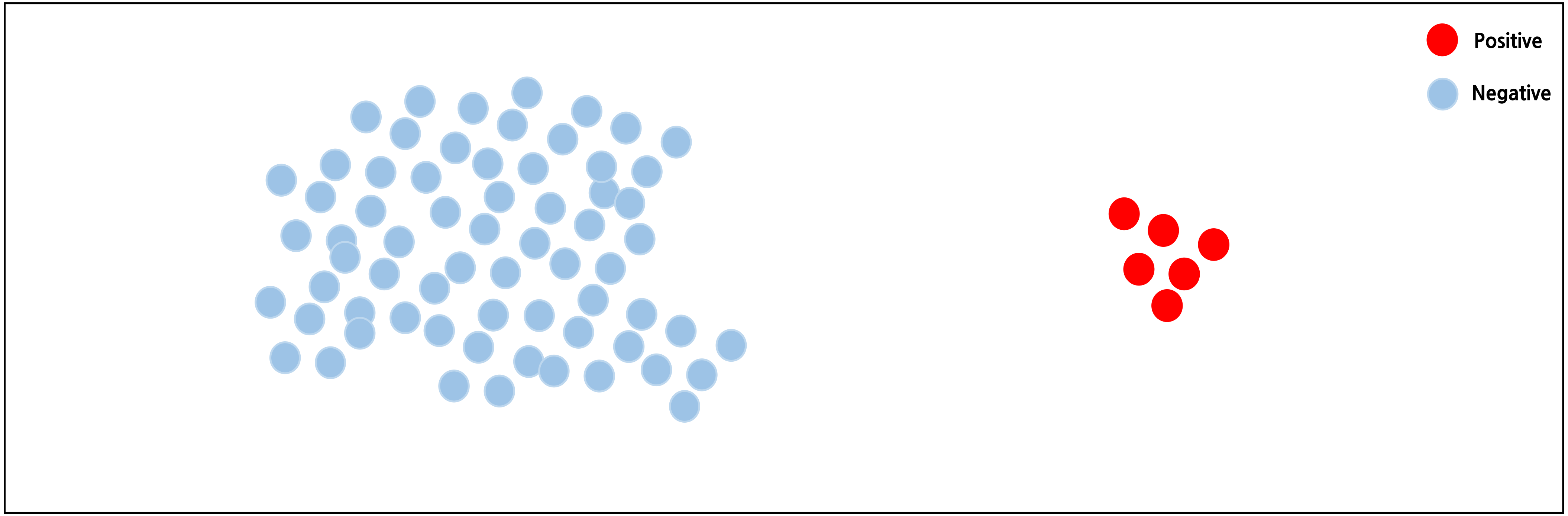
- Lots of feature selection techniques

# II. How to solve Class Imbalance problem

## ❖ Over Sampling

① How to create repeatedly instances of positive class.

② SMOTE : Synthetic Minority Over-Sampling



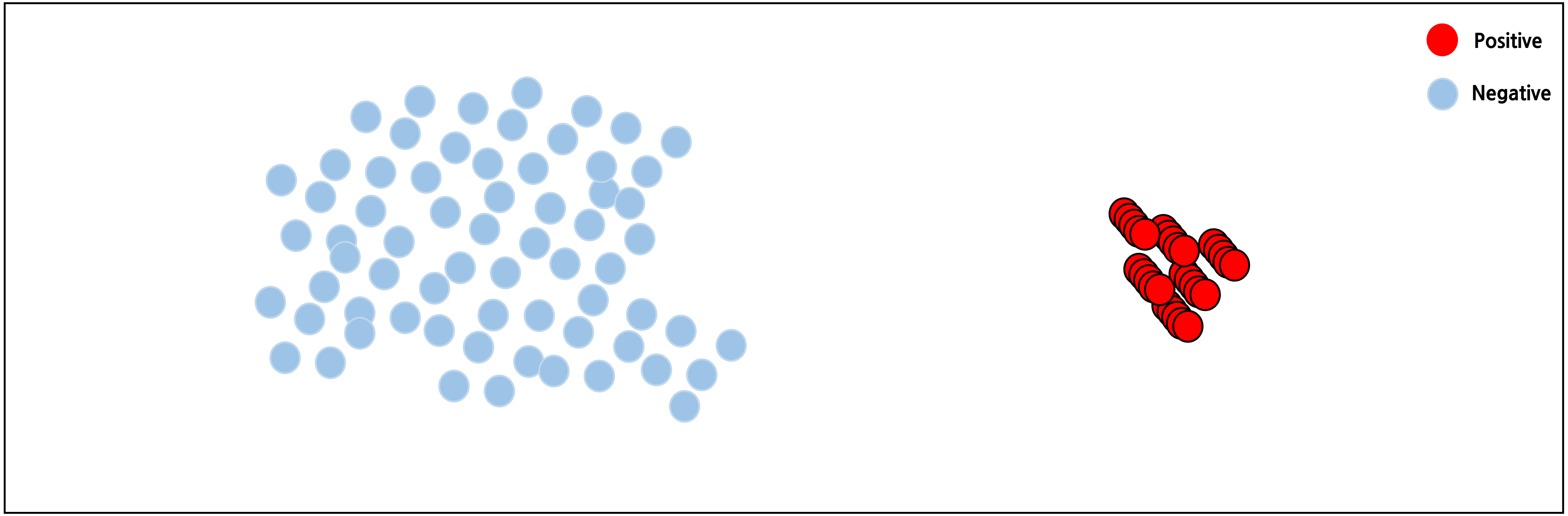

Feature Space

# Ⅱ. How to solve Class Imbalance problem

## ❖ Over Sampling

① How to create repeatedly instances of positive class.

② SMOTE : Synthetic Minority Over-Sampling



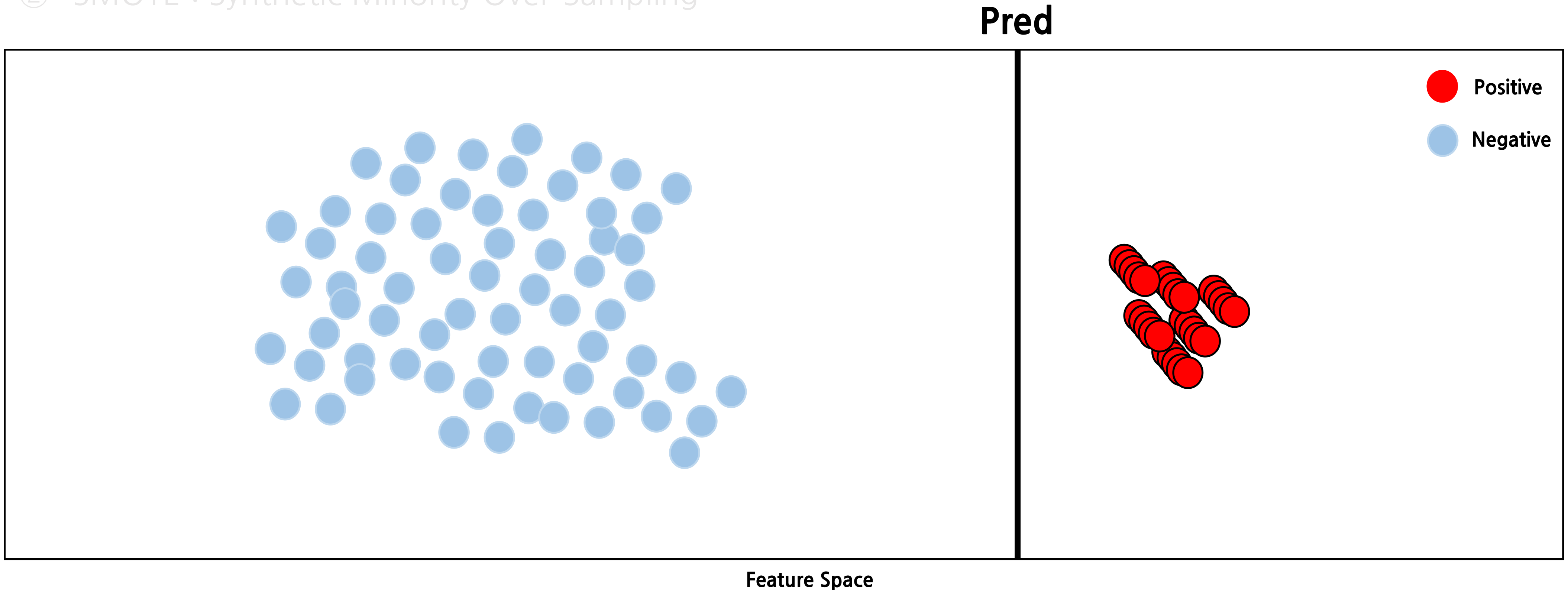

Feature Space

# II. How to solve Class Imbalance problem

## ❖ Over Sampling

① How to create repeatedly instances of positive class.

② SMOTE : Synthetic Minority Over-Sampling



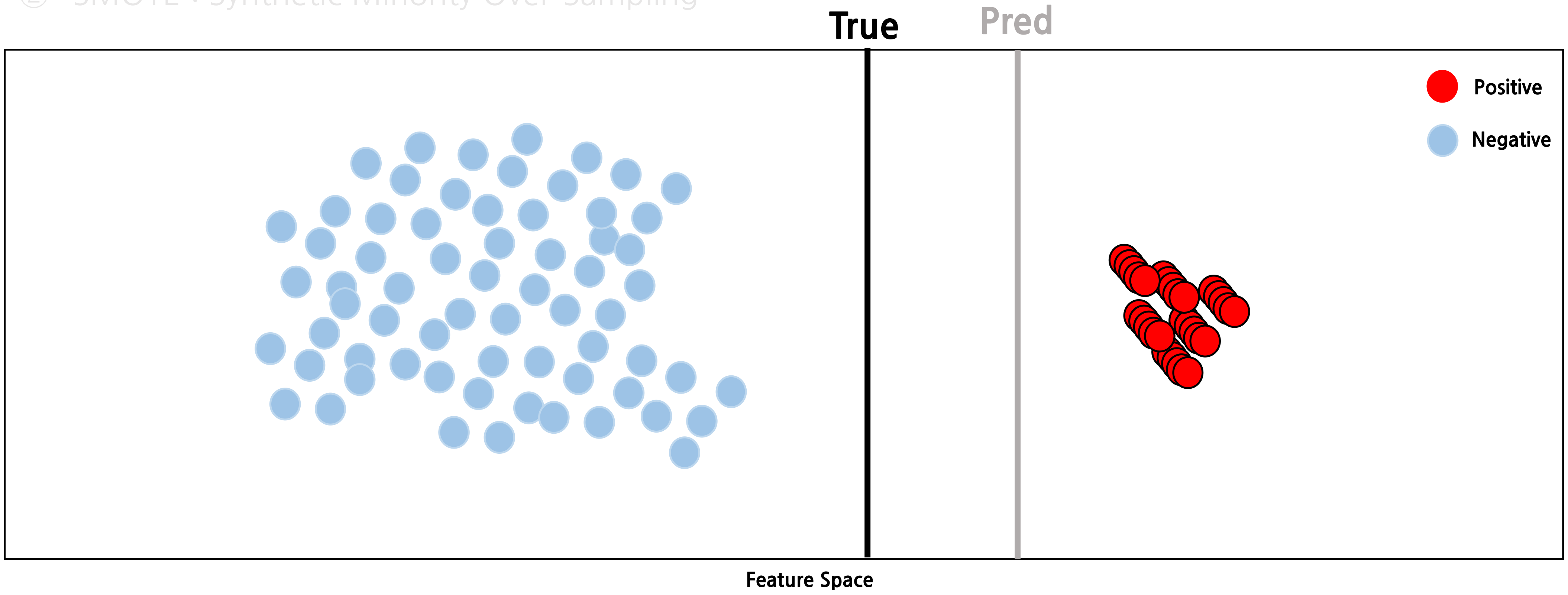

DMQA

# II. How to solve Class Imbalance problem

## ❖ Over Sampling

① How to create repeatedly instances of positive class.

② SMOTE : Synthetic Minority Over-Sampling



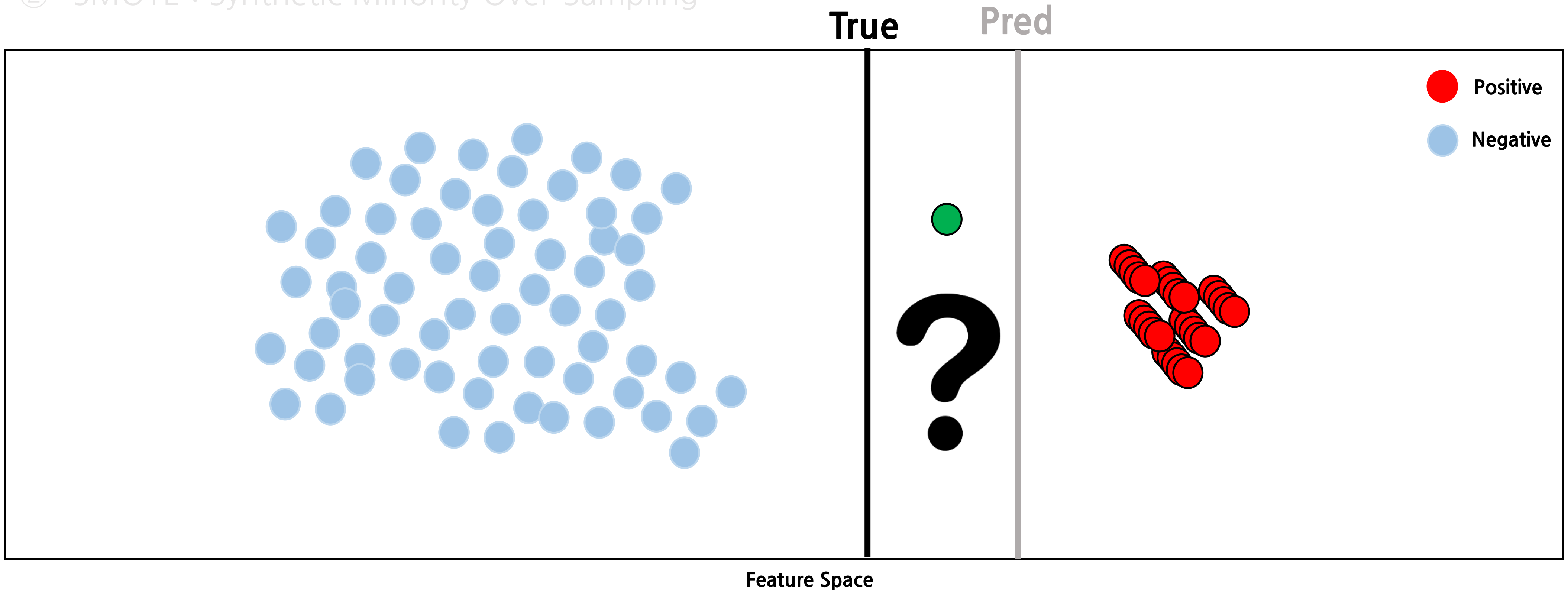

DMQA

# II. How to solve Class Imbalance problem

## ❖ Over Sampling

① How to create repeatedly instances of positive class.

② SMOTE : Synthetic Minority Over-Sampling

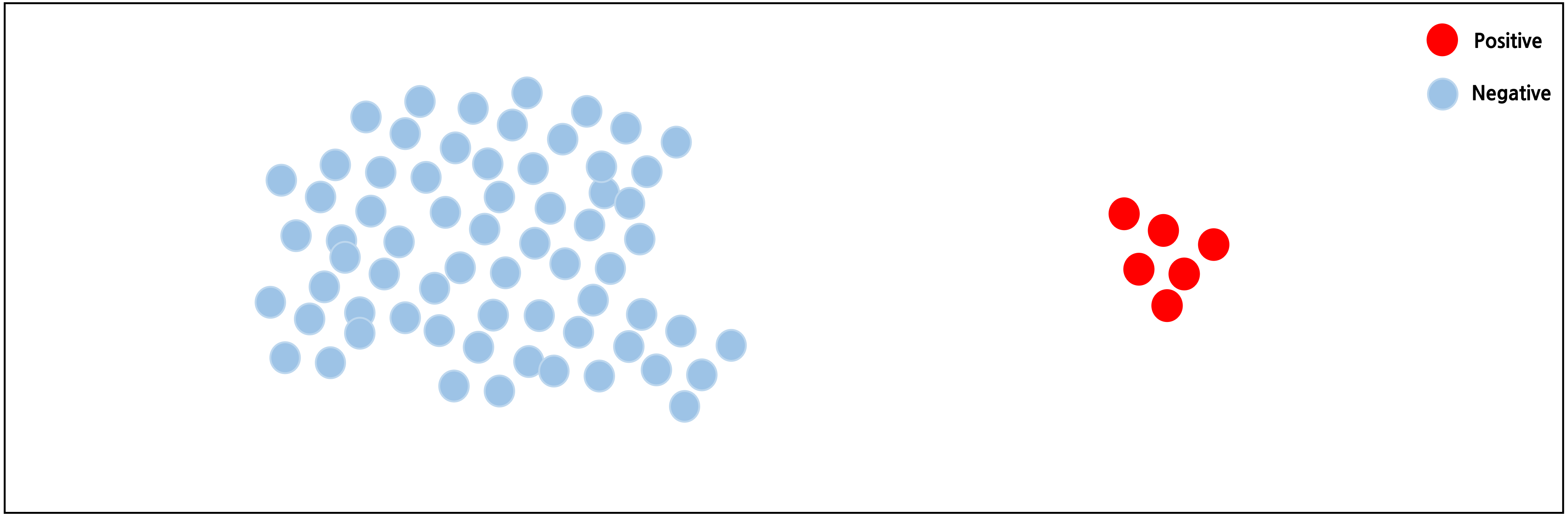


# II. How to solve Class Imbalance problem

## ❖ Over Sampling

① How to create repeatedly instances of positive class.

② SMOTE : Synthetic Minority Over-Sampling



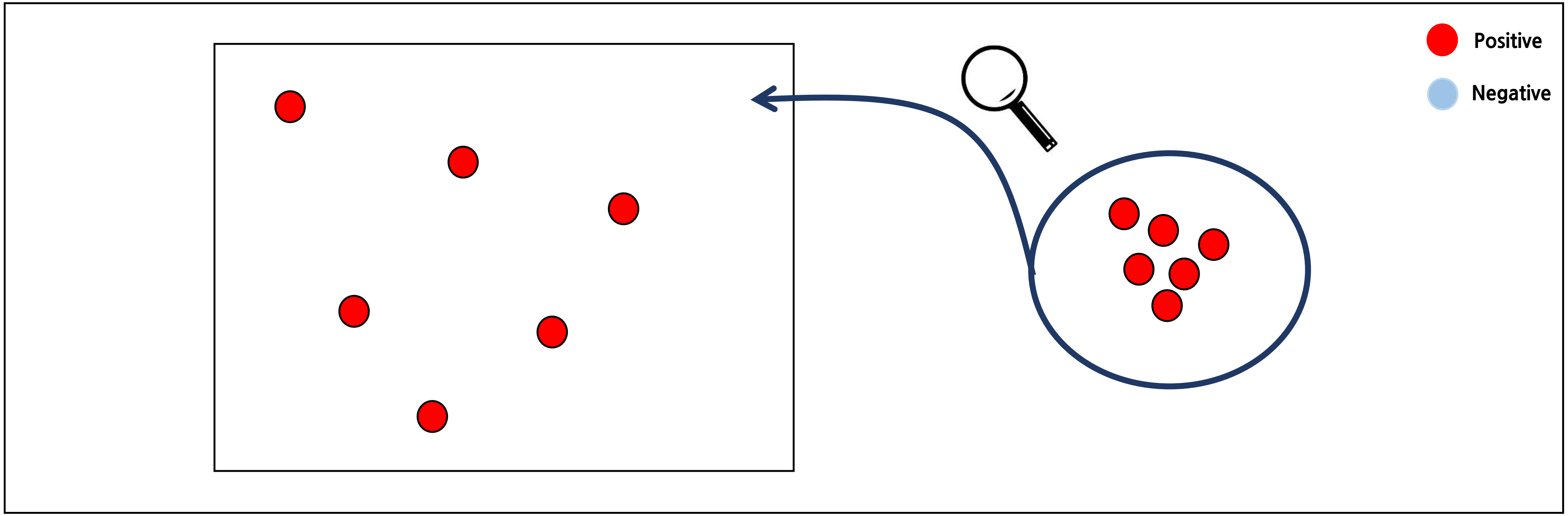

Feature Space

# II. How to solve Class Imbalance problem

## ❖ Over Sampling

① How to create repeatedly instances of positive class.

② SMOTE : Synthetic Minority Over-Sampling



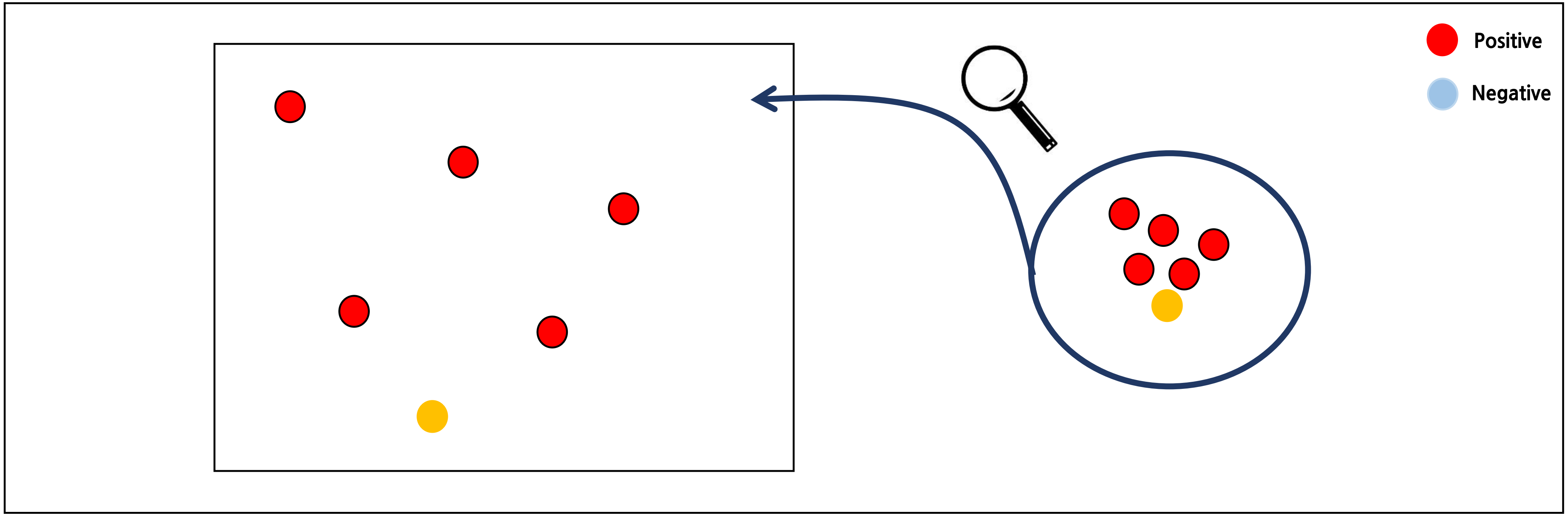

Feature Space

# II. How to solve Class Imbalance problem

## ❖ Over Sampling

### ② SMOTE : Synthetic Minority Over-Sampling

➢ Select a observation from the minority(positive) class.



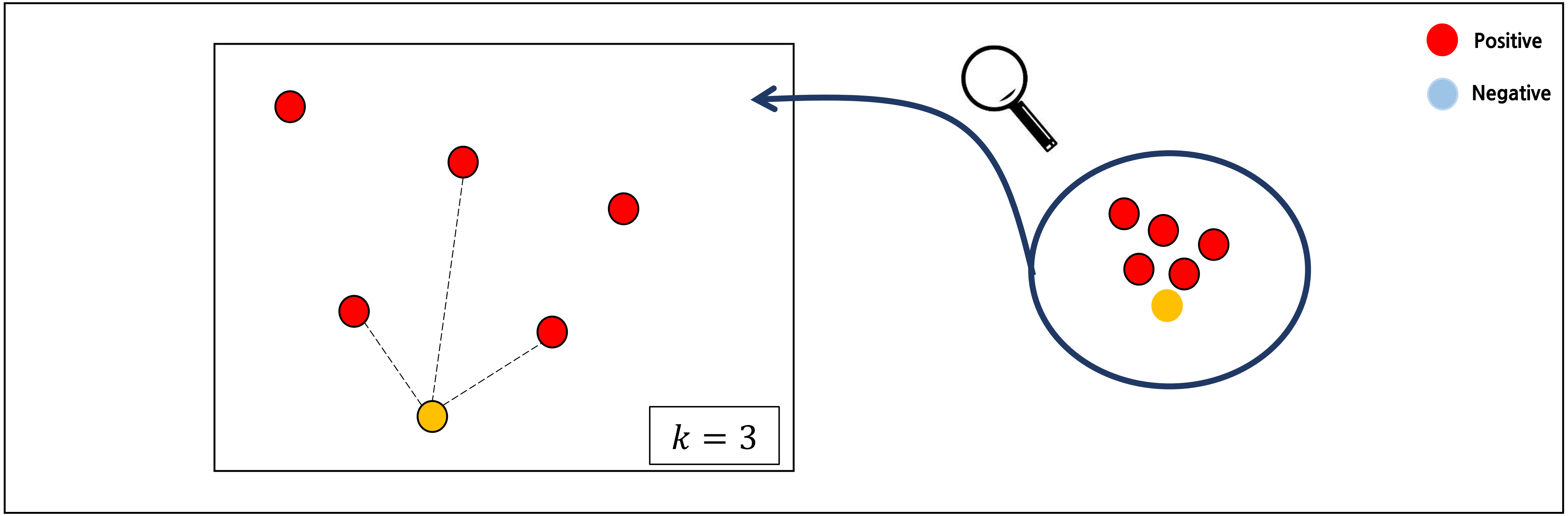

Feature Space

# II. How to solve Class Imbalance problem

## ❖ Over Sampling

### ② SMOTE : Synthetic Minority Over-Sampling

- Select $k$ nearst neighbors.($k$ = $hyperparameter$)



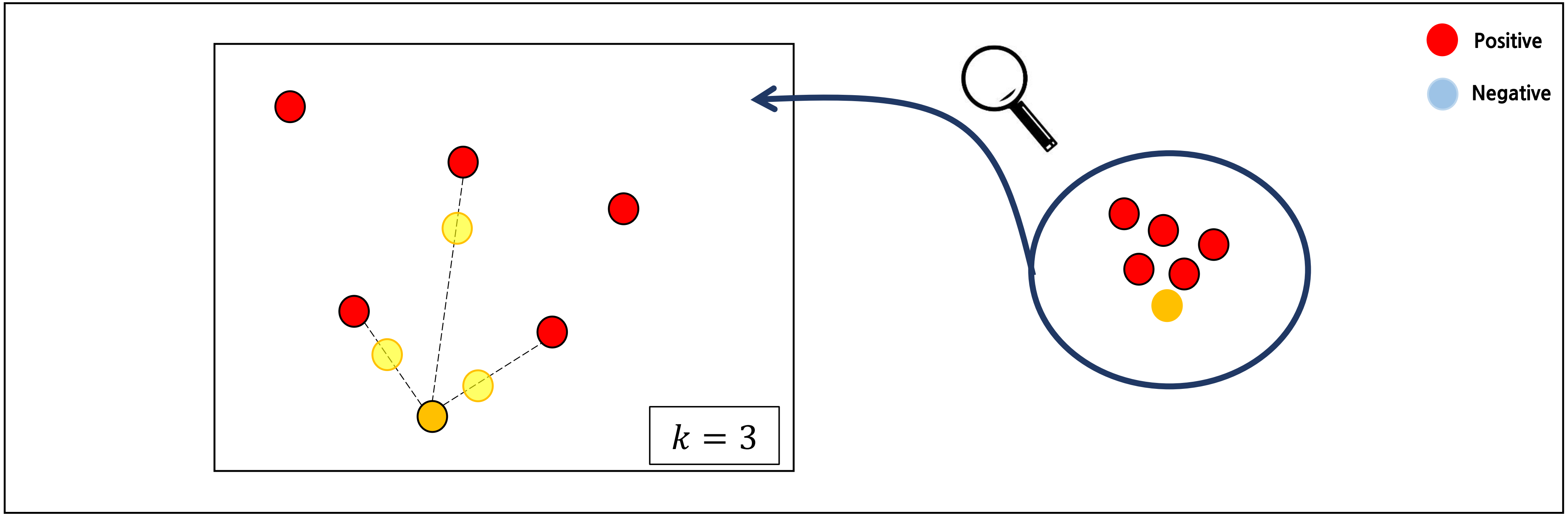

Feature Space

# II. How to solve Class Imbalance problem

## ❖ Over Sampling

### ② SMOTE : Synthetic Minority Over-Sampling

➢ Create a minority(positive) class arbitrarily in a straight line between two points.



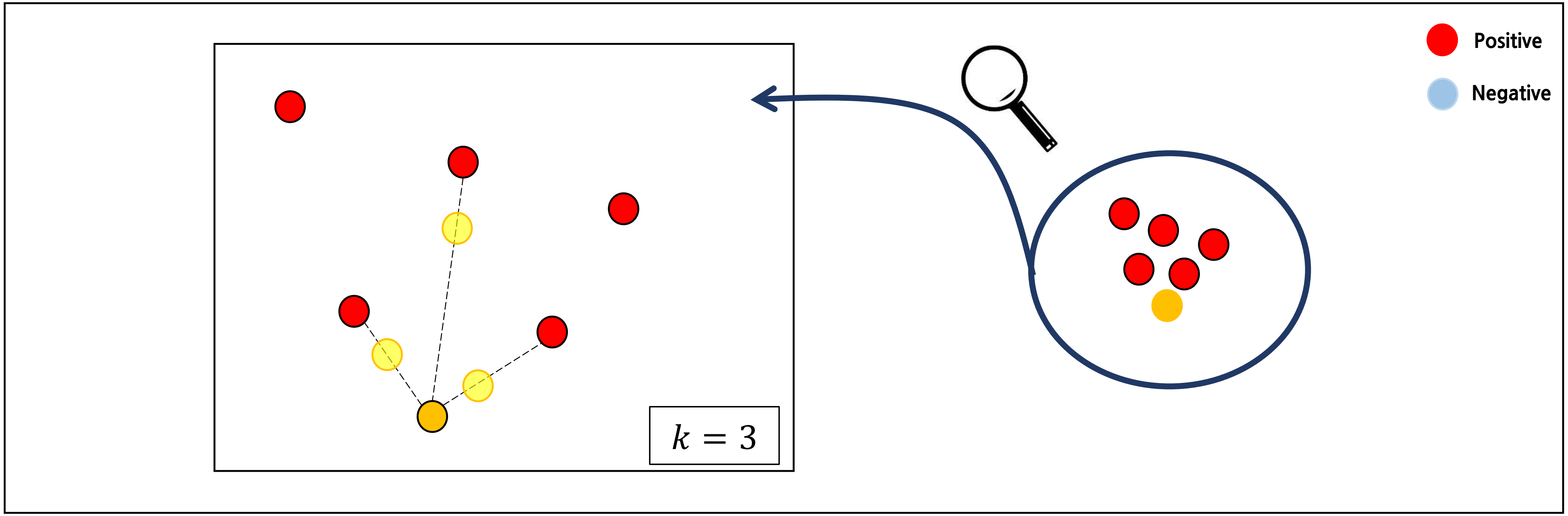

Feature Space

# II. How to solve Class Imbalance problem

## ❖ Over Sampling

### ② SMOTE : Synthetic Minority Over-Sampling

➢ Create a minority(positive) class arbitrarily in a straight line between two points. -> Interpolation



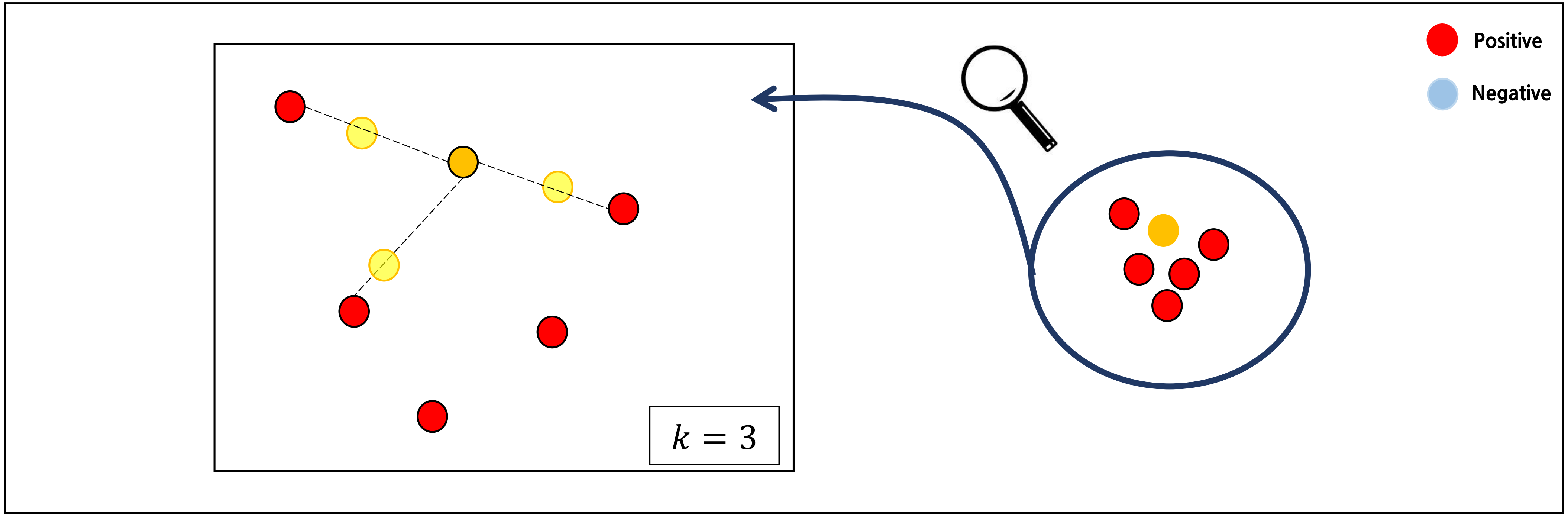

Feature Space

# II. How to solve Class Imbalance problem

## ❖ Over Sampling

### ② SMOTE : Synthetic Minority Over-Sampling

➢ Create a minority(positive) class arbitrarily in a straight line between two points. -> Interpolation



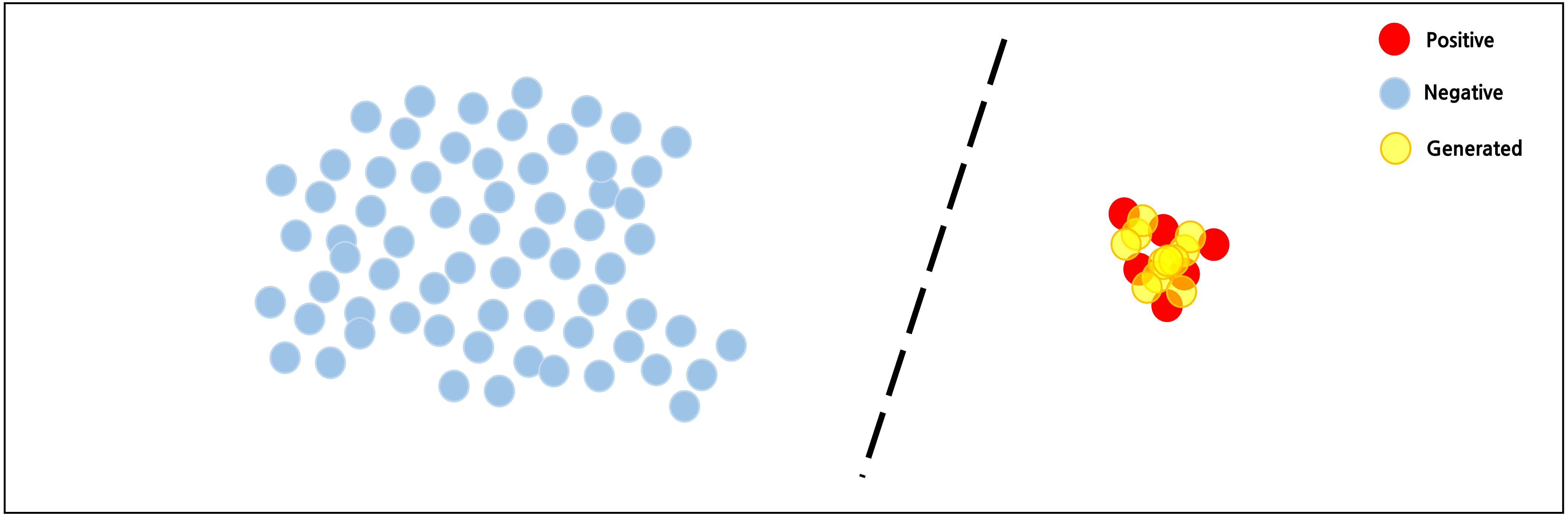

Feature Space

# II. How to solve Class Imbalance problem

## ❖ Over Sampling

### ② SMOTE : Synthetic Minority Over-Sampling

➢ After generation is complete, apply a classification algorithm.

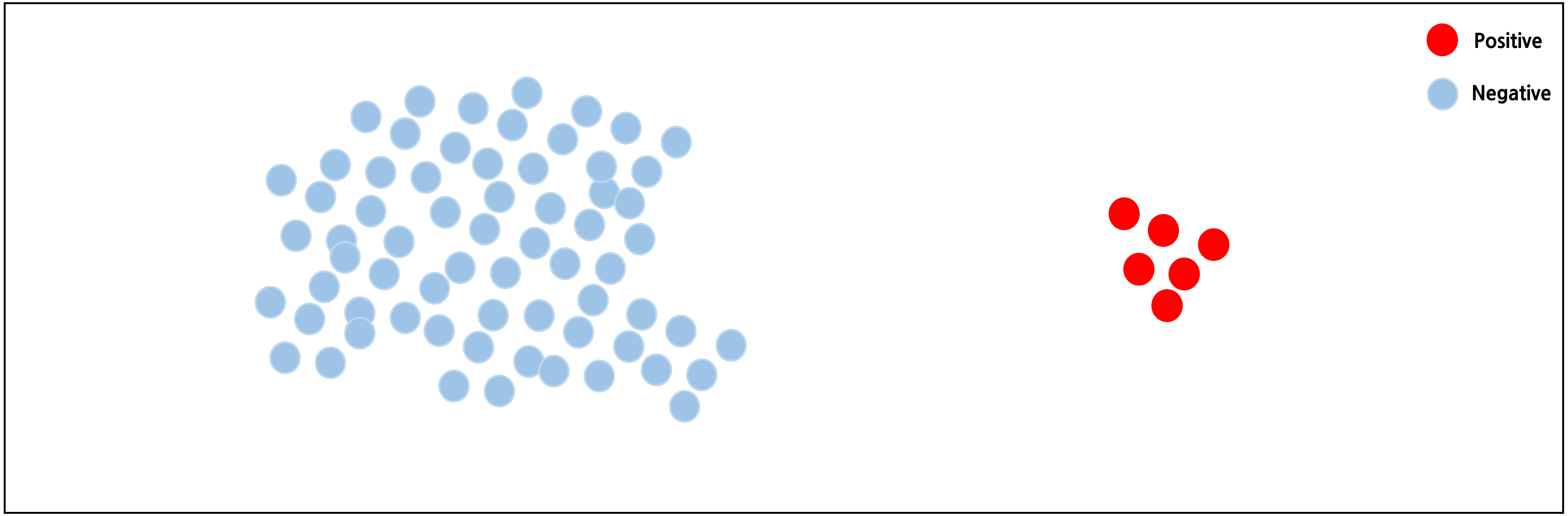



Feature Space

# II. How to solve Class Imbalance problem

## ❖ Under Sampling

① RUS : Random Under Sampling



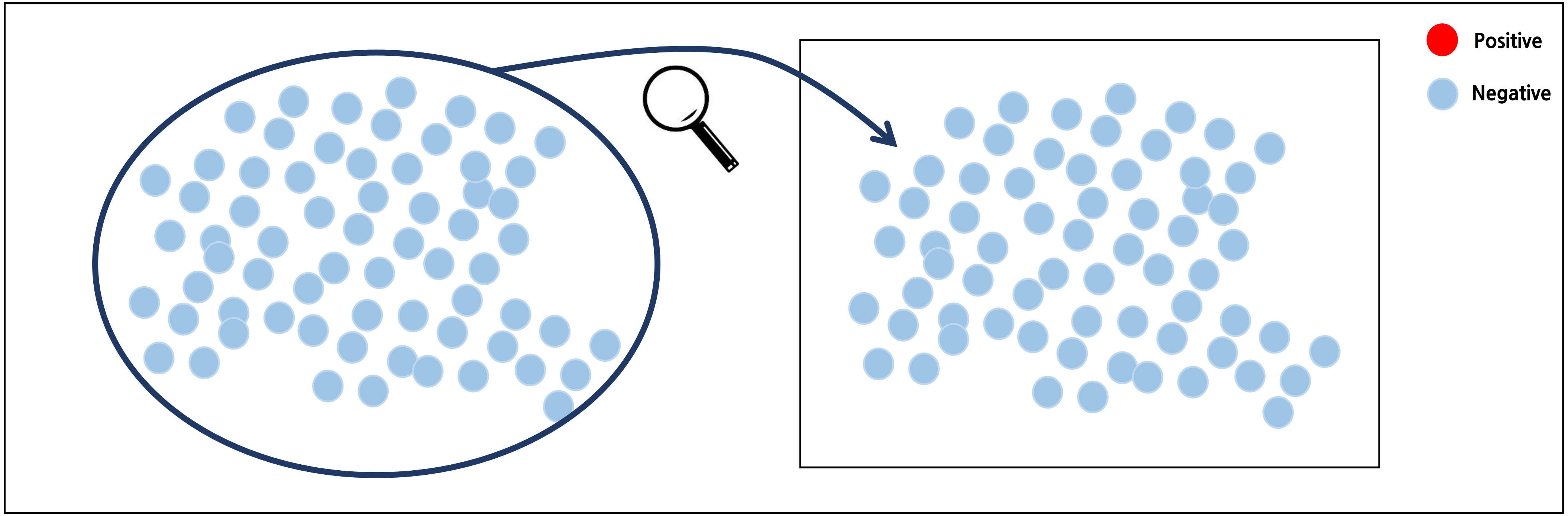

Feature Space

# II. How to solve Class Imbalance problem

## ❖ Under Sampling

① RUS : Random Under Sampling

➢ Remove negative(majority) observations randomly.



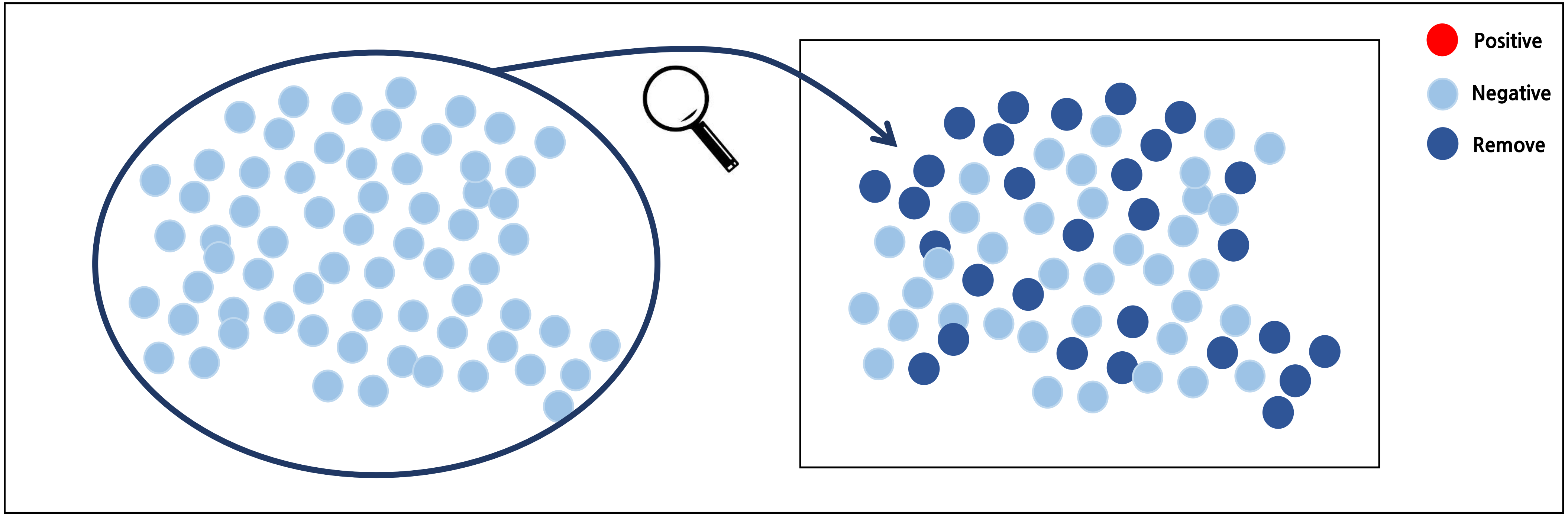

Feature Space

# II. How to solve Class Imbalance problem

## ❖ Under Sampling

① RUS : Random Under Sampling

- ➢ Remove negative(majority) observations randomly.



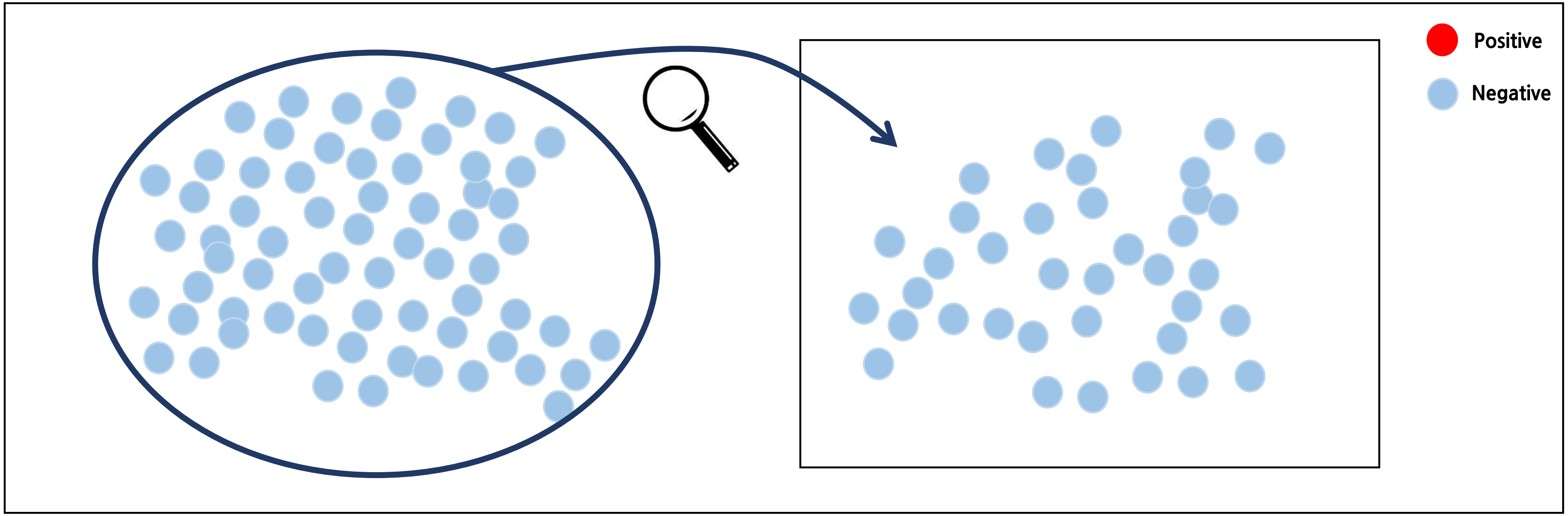

Feature Space

# II. How to solve Class Imbalance problem

## ❖ Under Sampling

① RUS : Random Under Sampling

➢ Remove negative(majority) observations randomly.




Feature Space

# II. How to solve Class Imbalance problem

## ❖ Comparison Under sampling with Over sampling

| | Advantages | Disadvantages |
|---|---|---|
| Under Sampling | ① We reduce the size of the training dataset by removing the data from the negative (majority) class.<br><br>② **Time to train model when using under sampling techniques is shorter than oversampling techniques.** | ① Because we remove observations, we can not use the information that we have in the modeling process. |
| Over Sampling | ① Since observations are not removed, no loss of information occurs.<br><br>② Because of the use of interpolation, class boundaries do not change. **That is, the distribution of the positive (minority) class does not change.** | ① **Because it creates observations for the positive (minority) class, it takes larger time to train the training data than under sampling.** |

# II. How to solve Class Imbalance problem

❖ Sampling Techniques

- Over Sampling vs. Under Sampling

❖ Algorithm Techniques

- Boosting, AdaBoost,……

❖ Feature selection Techniques

- Lots of feature selection techniques

# II. How to solve Class Imbalance problem

## ❖ Boosting

- **Boosting is an ensemble method** that creates a predictive model by continuously building weak models to better classify misclassified observations.

# II. How to solve Class Imbalance problem

## ❖ Boosting

- Boosting is an ensemble method that creates a predictive model by continuously building weak models to better classify misclassified observations.

- It takes a long time to generate a weak classifier based on misclassified observations, but it performs better than normal classifiers(ex. Decision Tree, logistic regression).

# II. How to solve Class Imbalance problem

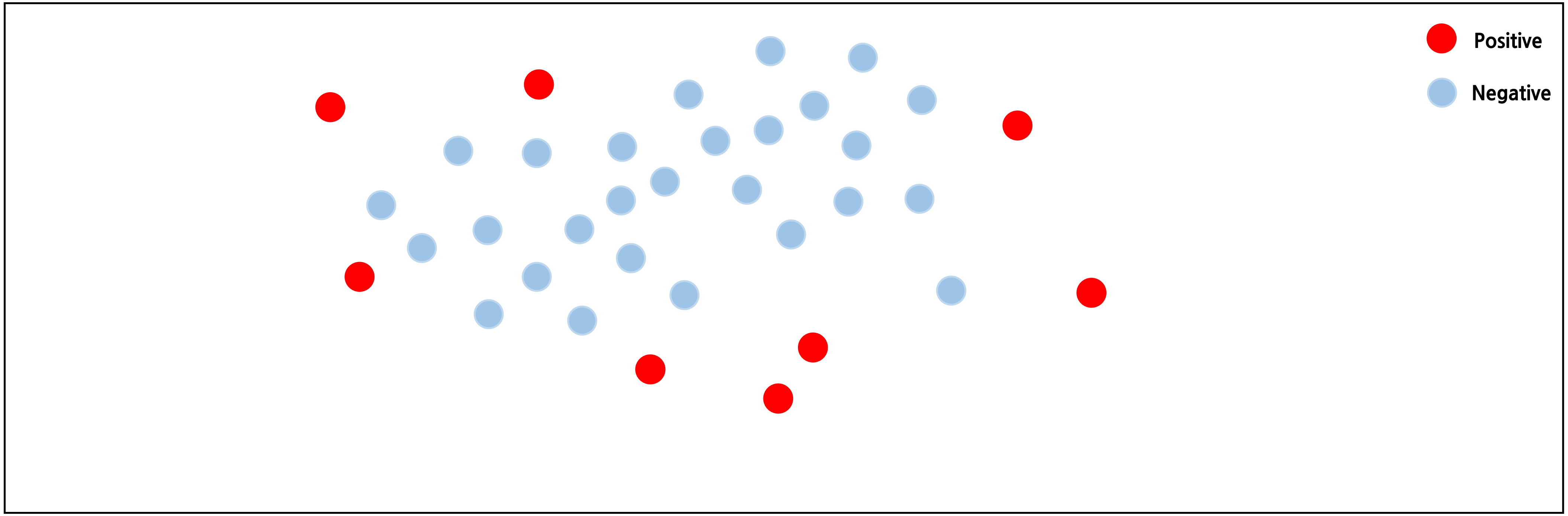
## ❖ AdaBoost(Adaptive Boosting)

- **AdaBoost** is a boosting method that creates weak classifiers while giving **larger weight** to misclassified observations than well-classified observations.

# II. How to solve Class Imbalance problem

## ❖ AdaBoost(Adaptive Boosting)

- **AdaBoost** is a boosting method that creates weak classifiers while giving **larger weight** to misclassified observations than well-classified observations.

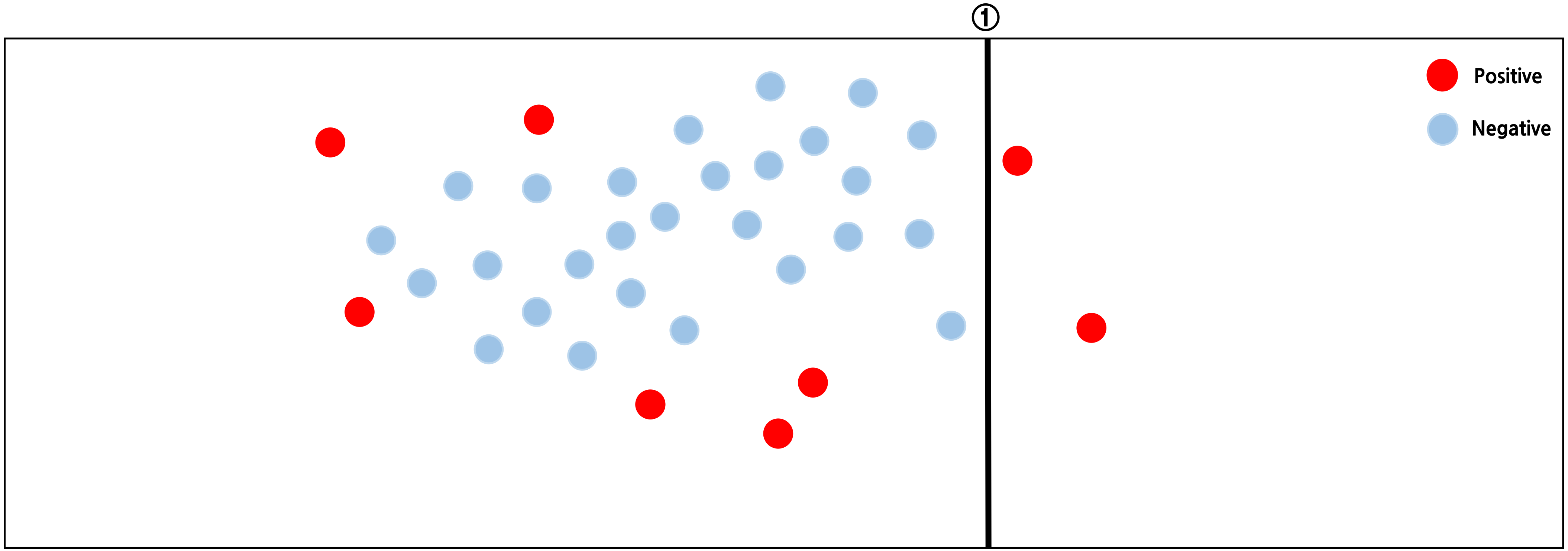



Feature Space

# II. How to solve Class Imbalance problem

## ❖ AdaBoost(Adaptive Boosting)

- **AdaBoost** is a boosting method that creates weak classifiers while giving **larger weight** to misclassified observations than well-classified observations.

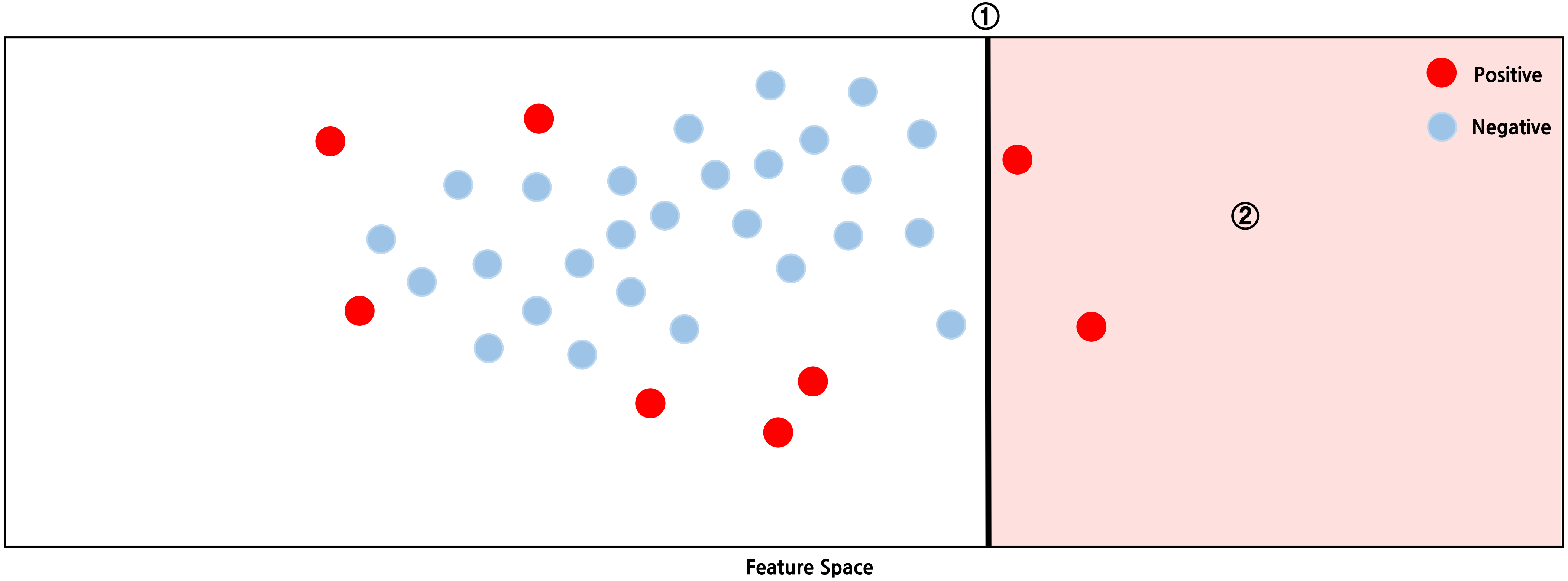



Feature Space

# II. How to solve Class Imbalance problem

## ❖ AdaBoost(Adaptive Boosting)

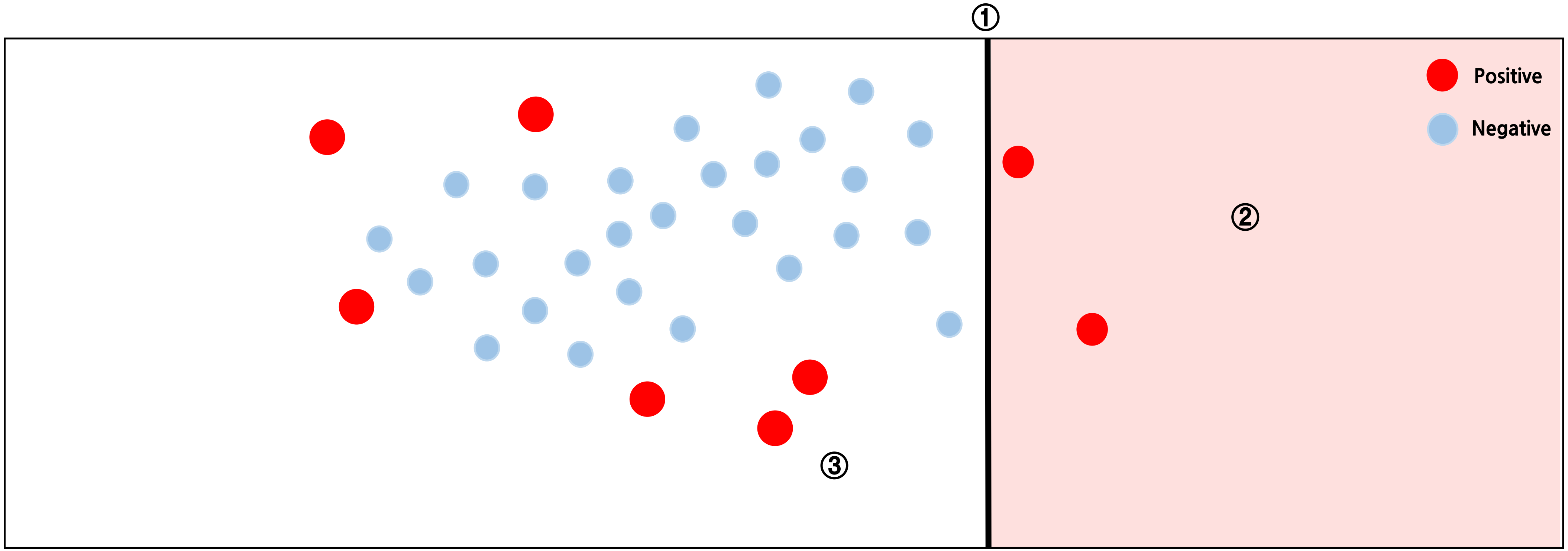
- **AdaBoost** is a boosting method that creates weak classifiers while giving **larger weight** to misclassified observations than well-classified observations.



# II. How to solve Class Imbalance problem

## ❖ AdaBoost(Adaptive Boosting)

- **AdaBoost** is a boosting method that creates weak classifiers while giving **larger weight** to misclassified observations than well-classified observations.

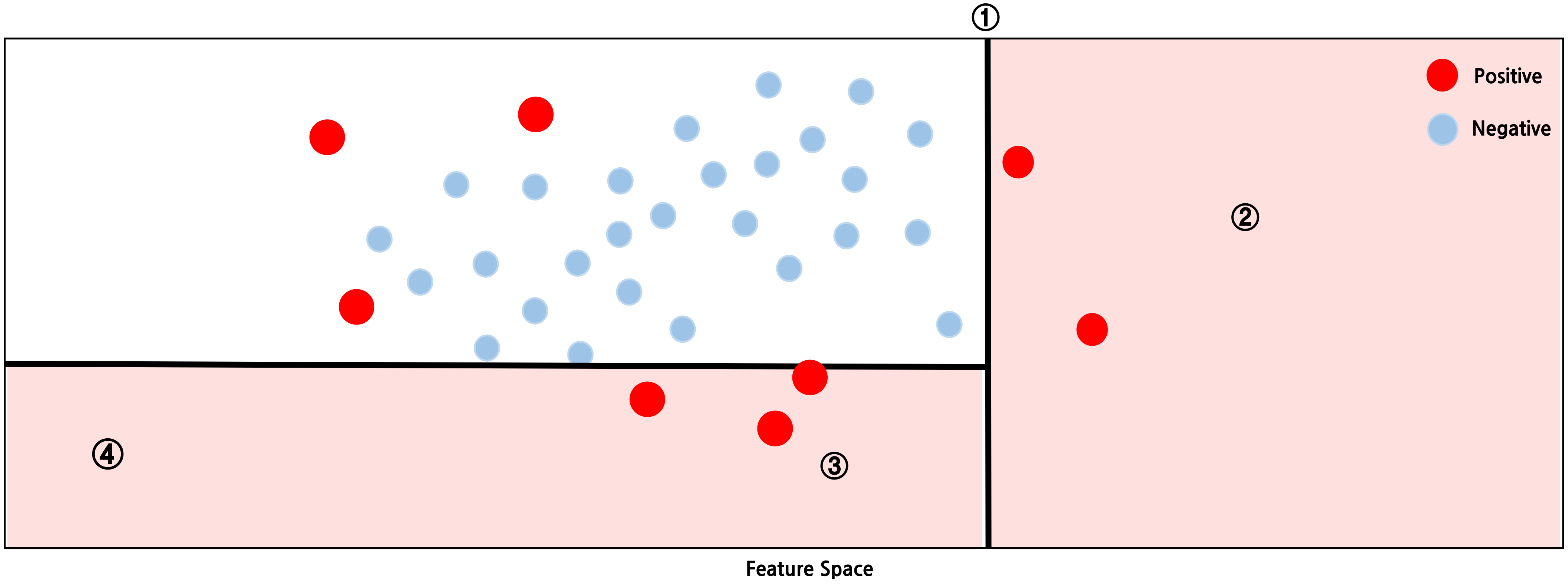



Feature Space

# II. How to solve Class Imbalance problem

## ❖ AdaBoost(Adaptive Boosting)

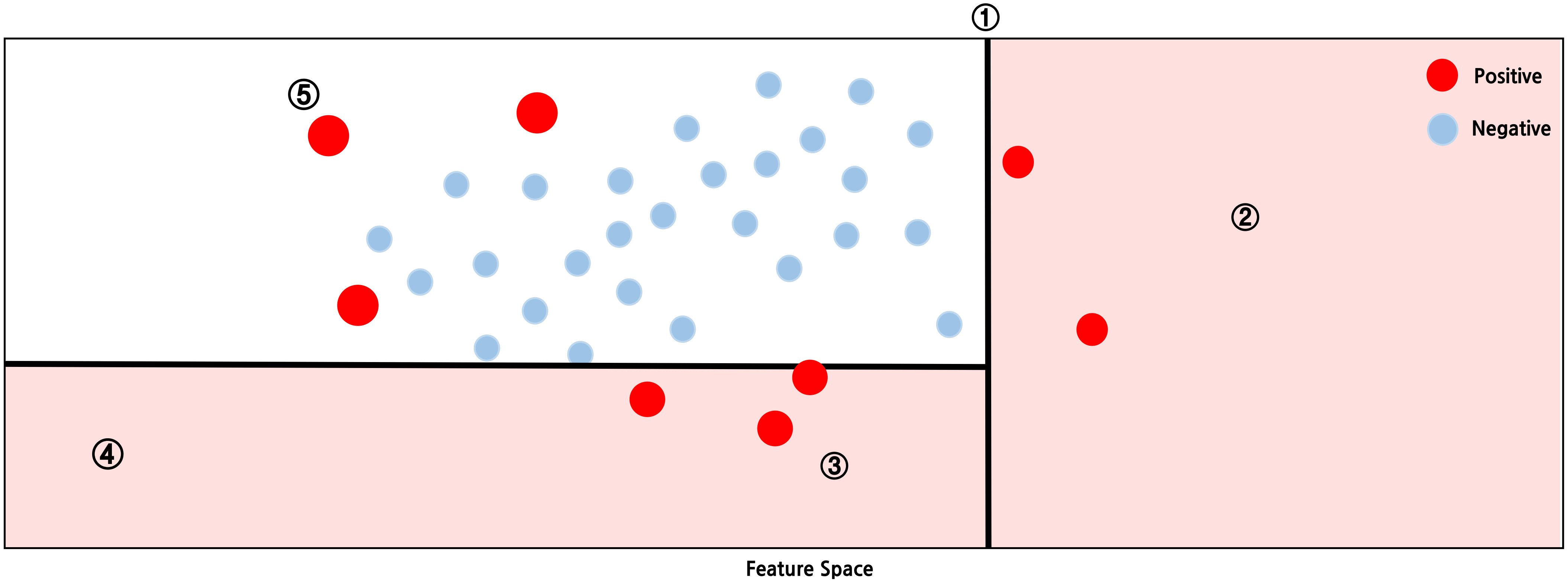
- **AdaBoost** is a boosting method that creates weak classifiers while giving **larger weight** to misclassified observations than well-classified observations.



# II. How to solve Class Imbalance problem

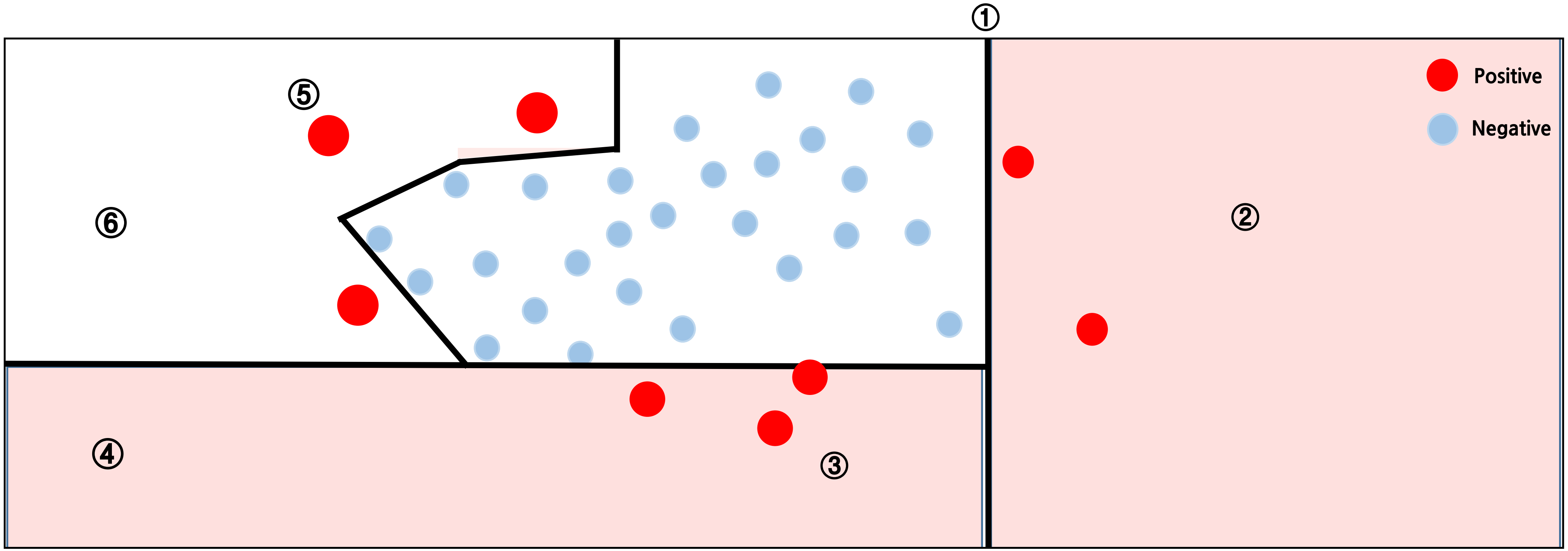
## ❖ AdaBoost(Adaptive Boosting)

- **AdaBoost** is a boosting method that creates weak classifiers while giving **larger weight** to misclassified observations than well-classified observations.



## ❖ AdaBoost(Adaptive Boosting)

- **AdaBoost** is a boosting method that creates weak classifiers while giving **larger weight** to misclassified observations than well-classified observations.

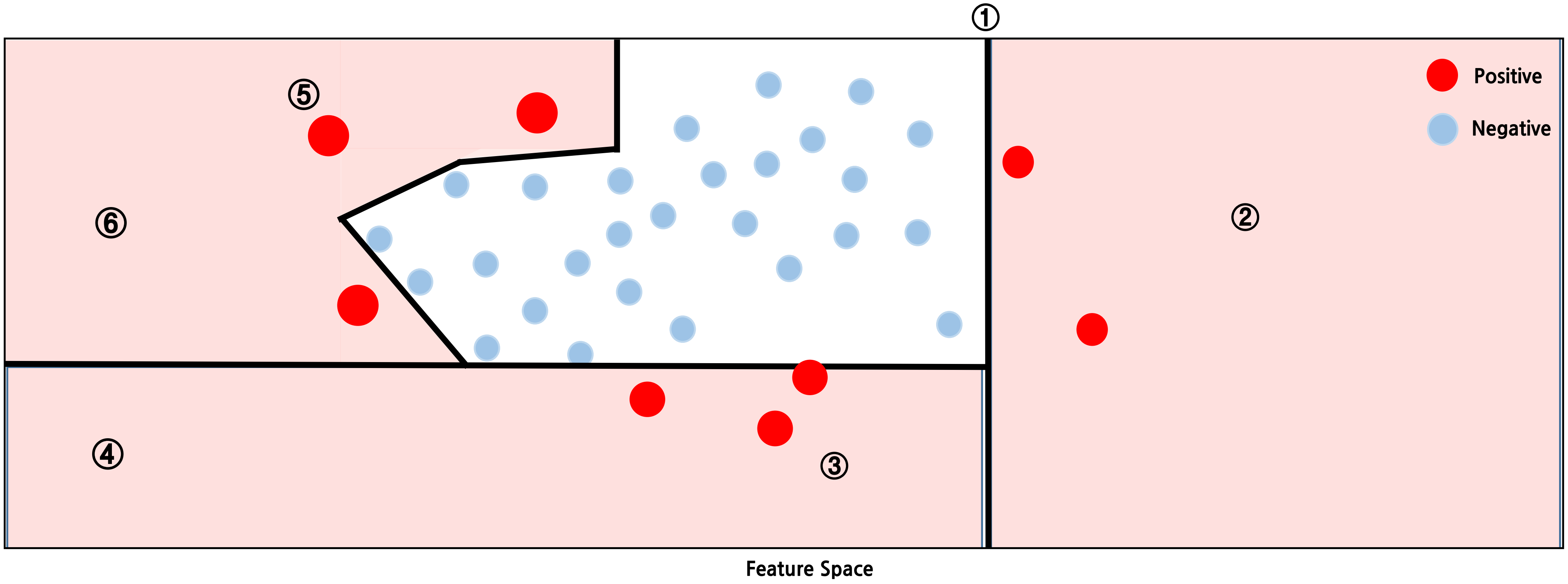



Feature Space

# II. How to solve Class Imbalance problem

## ❖ AdaBoost(Adaptive Boosting)

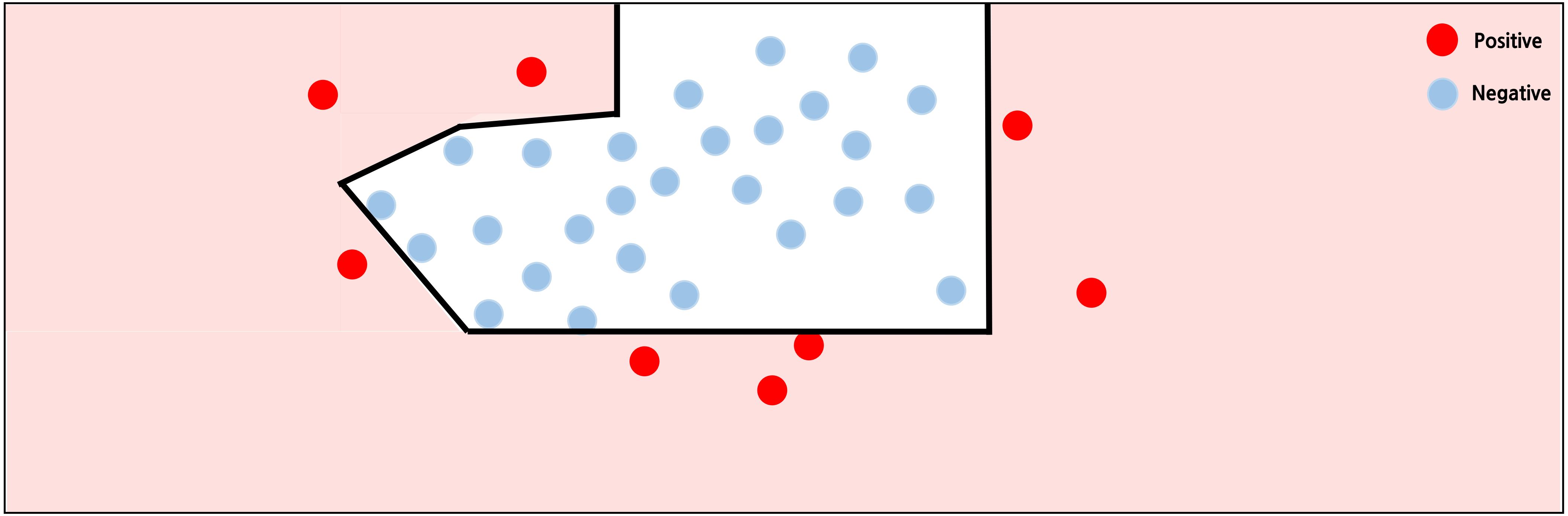
- **AdaBoost** is a boosting method that creates weak classifiers while giving **larger weight** to misclassified observations than well-classified observations.



## II. How to solve Class Imbalance problem

### ❖ AdaBoost(Adaptive Boosting)

- **AdaBoost** is a boosting method that creates weak classifiers while giving **larger weight** to misclassified observations than well-classified observations.




Feature Space

# II. How to solve Class Imbalance problem

***Procedure: AdaBoost Learning Algorithm***

1 ***Given***: $(x_1, y_1)\cdots(x_m, y_m)$, $x_i \in X$ , $y_i \in \{-1,1\}$

2 ***Initialize weight*** $D_1(i) = \frac{1}{m}$

3 ***For*** $t = 1 \cdots T$

***Call weak learn which returns weak classifier*** $h_t : X \in \{-1,1\}$ with minimum error w.r.t $D_t$

***Choose*** $\alpha_t \in R$

***Update weight***

$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

When $Z_t$ is a normalization factor chosen so that $D_{t+1}$ is a distribution

4 ***Produce the strong classifier***:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

# Contents

# III. RUSBoost vs. SMOTEBoost

## ❖ RUSBoost : A Hybrid Approach to Alleviating Class Imbalance

- IEEE TSMC(IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS)

**RUSBoost**: A **hybrid** approach to alleviating class imbalance
C Seiffert, TM Khoshgoftaar... - ... on Systems, Man ..., 2009 - ieeexplore.ieee.org
Class imbalance is a problem that is common to many application domains. When examples of one class in a training data set vastly outnumber examples of the other class (es), traditional data mining algorithms tend to create suboptimal classification models. Several ...

# III. RUSBoost vs. SMOTEBoost

?

❖ RUSBoost : A Hybrid Approach to Alleviating Class Imbalance

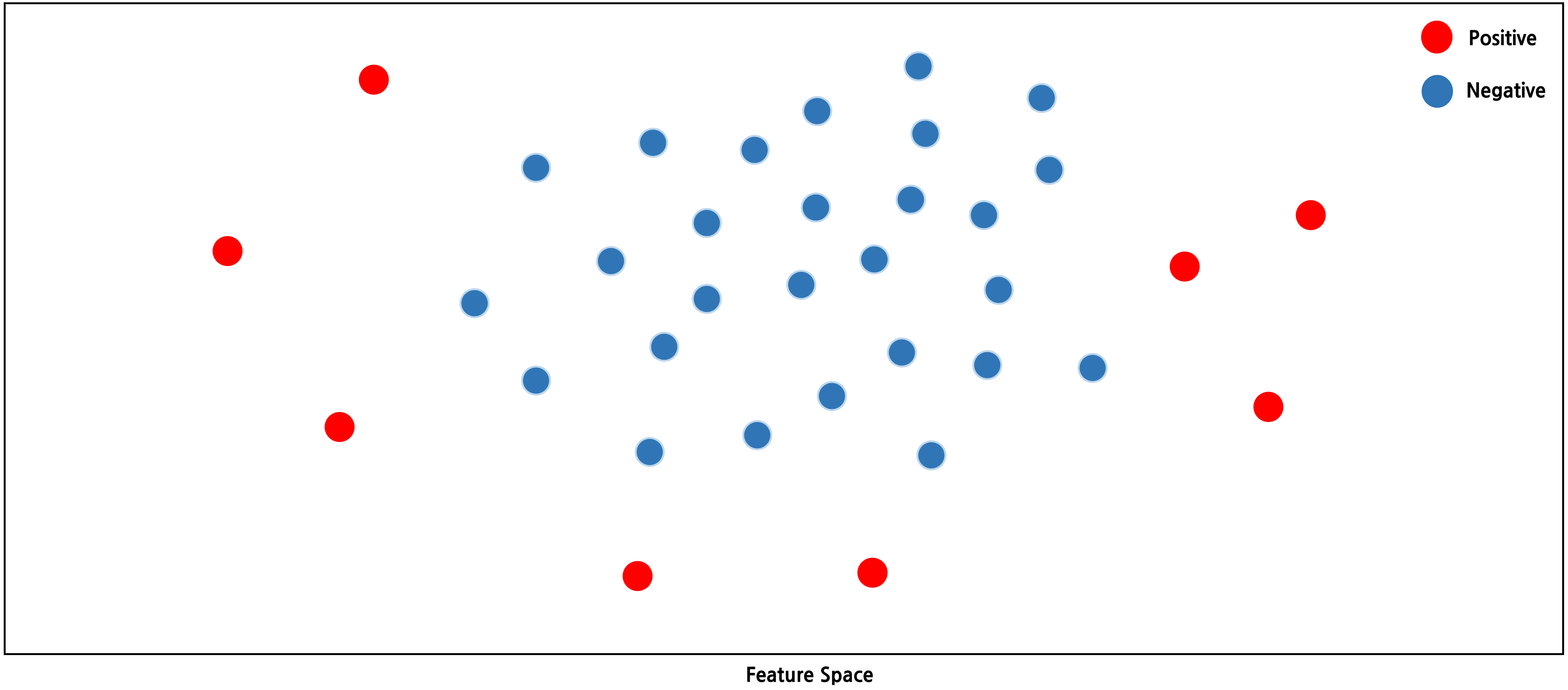
# Ⅲ. RUSBoost vs. SMOTEBoost

❖ RUSBoost : A Hybrid Approach to Alleviating Class Imbalance

**Hybrid = (Sampling + Algorithm) technique**

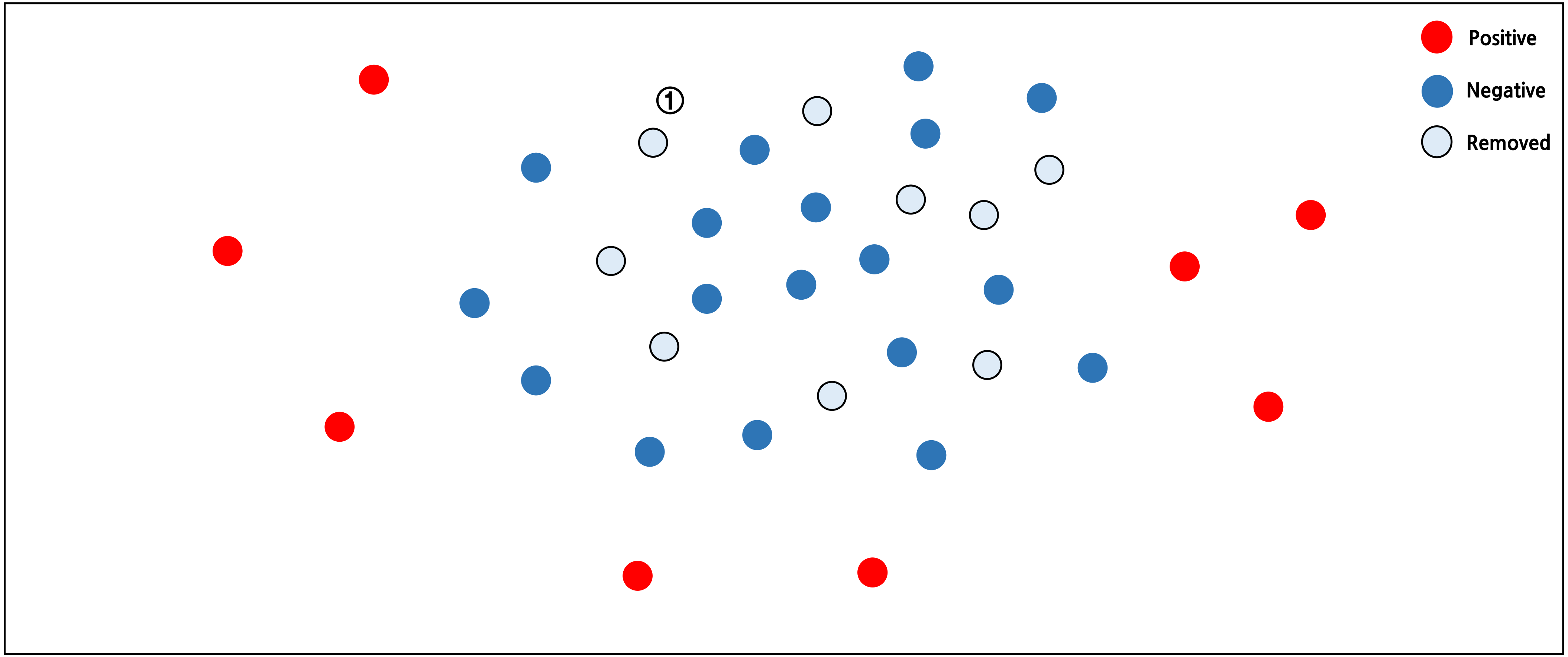
# III. RUSBoost vs. SMOTEBoost

❖ RUSBoost(RUS + AdaBoost)



# Ⅲ. RUSBoost vs. SMOTEBoost

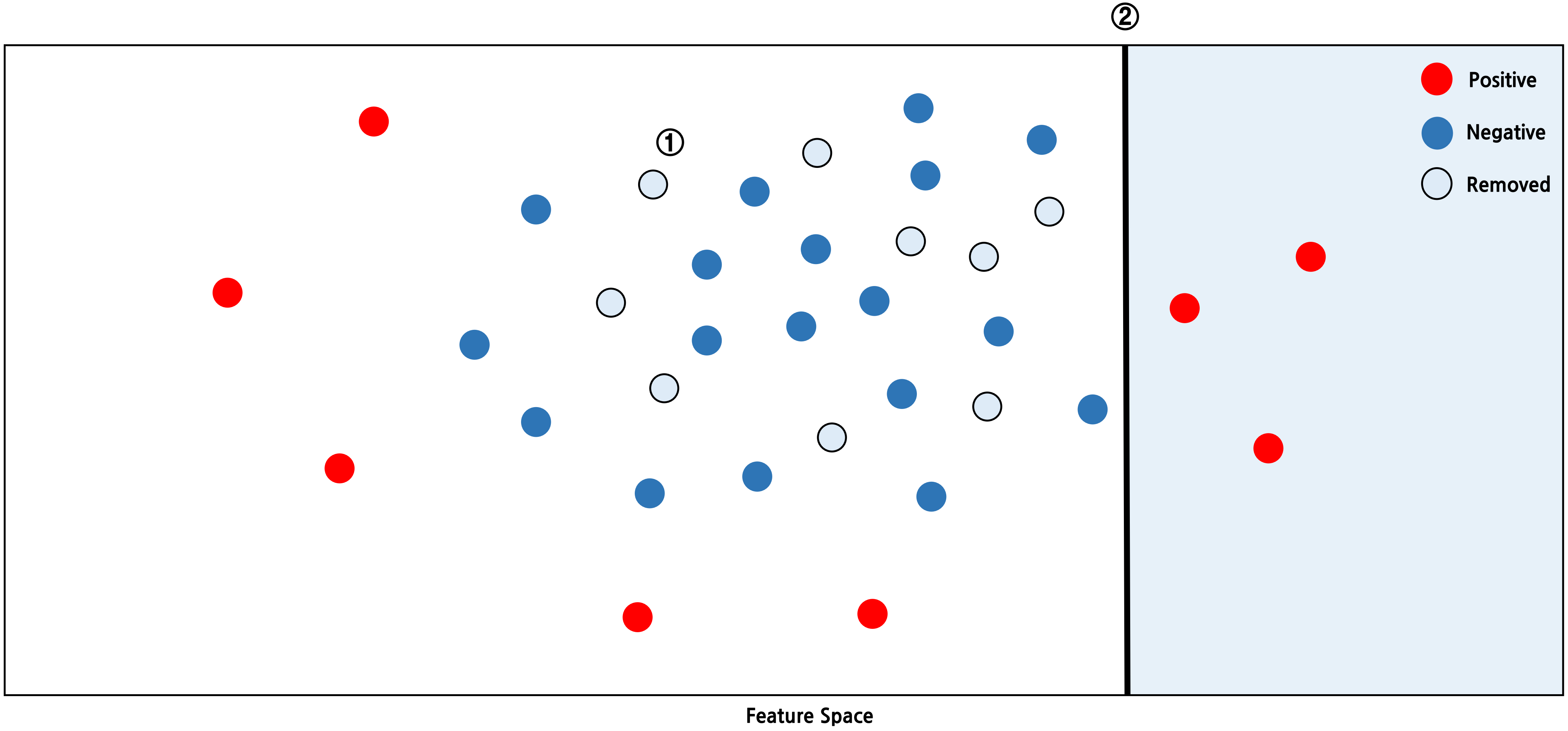
❖ RUSBoost(RUS + AdaBoost)




Feature Space

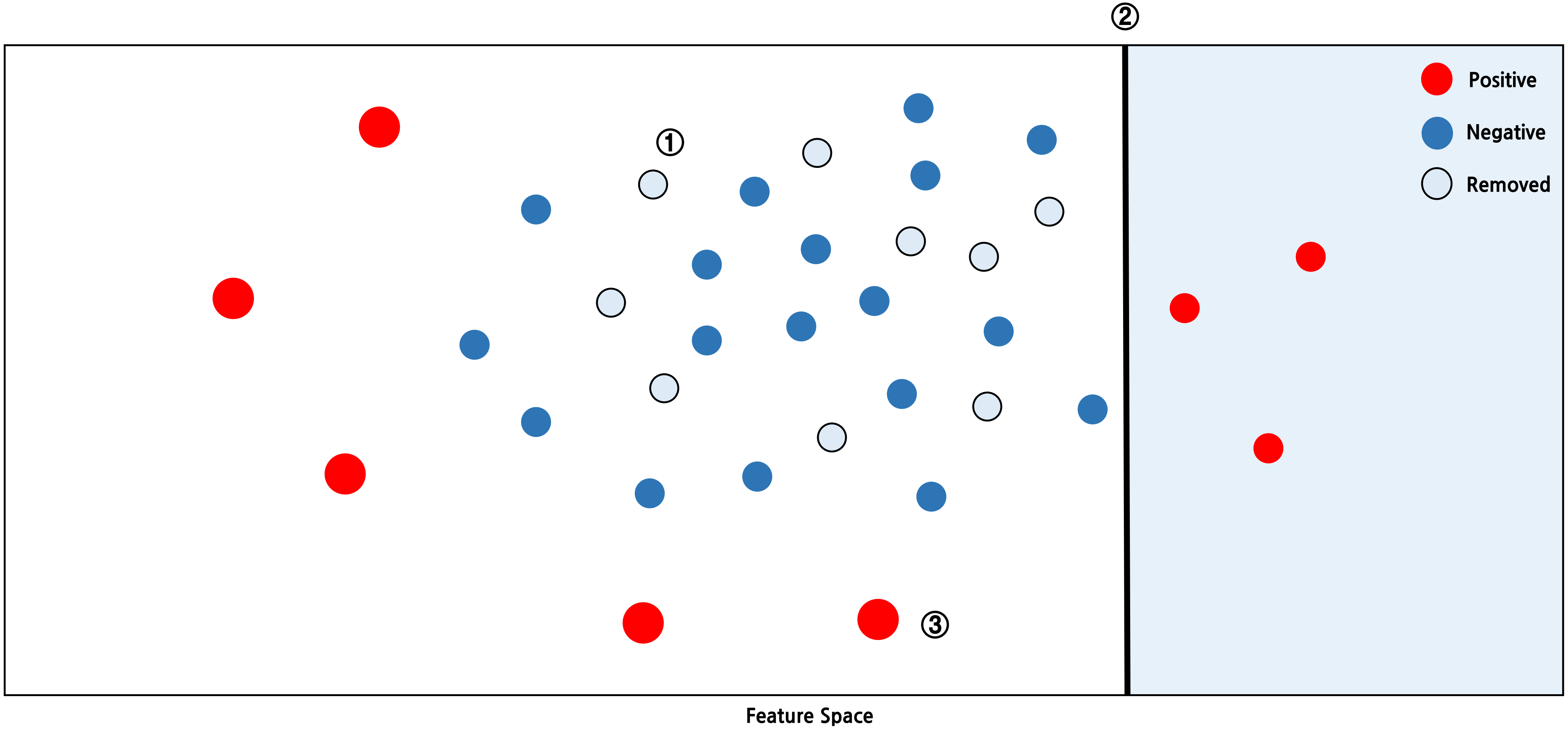
# Ⅲ. RUSBoost vs. SMOTEBoost

❖ RUSBoost(RUS + AdaBoost)



# III. RUSBoost vs. SMOTEBoost

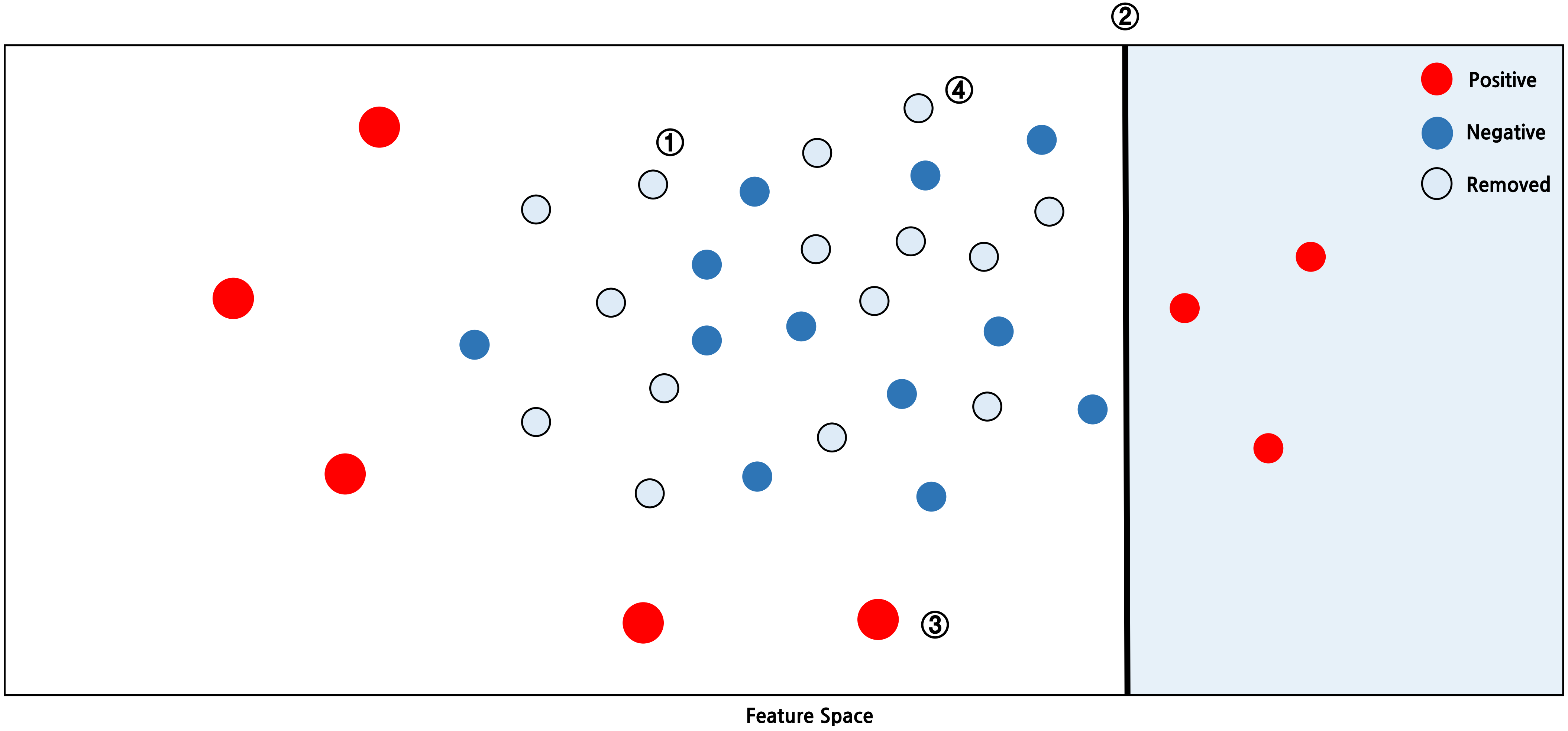
❖ RUSBoost(RUS + AdaBoost)




DMQA

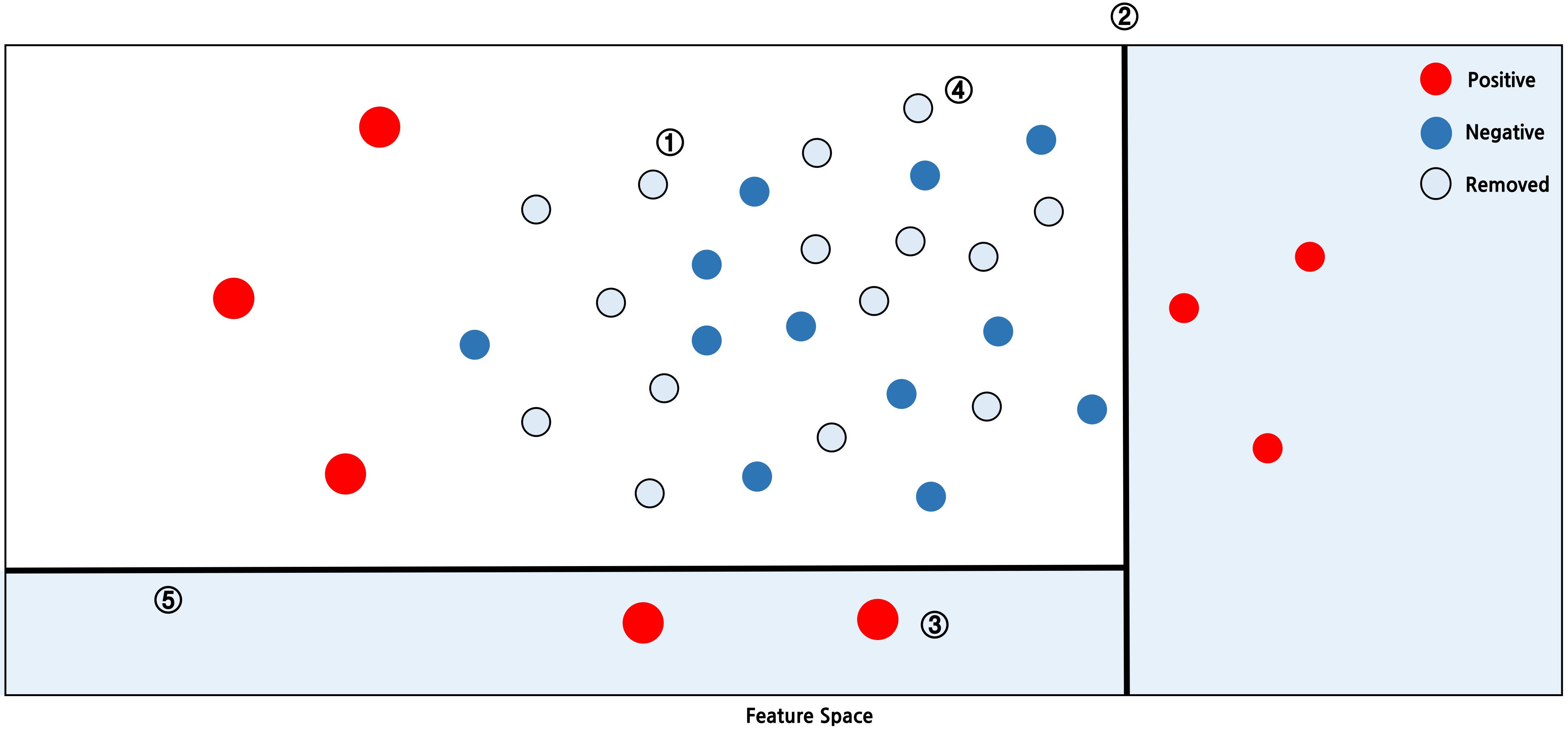
# Ⅲ. RUSBoost vs. SMOTEBoost

❖ RUSBoost(RUS + AdaBoost)



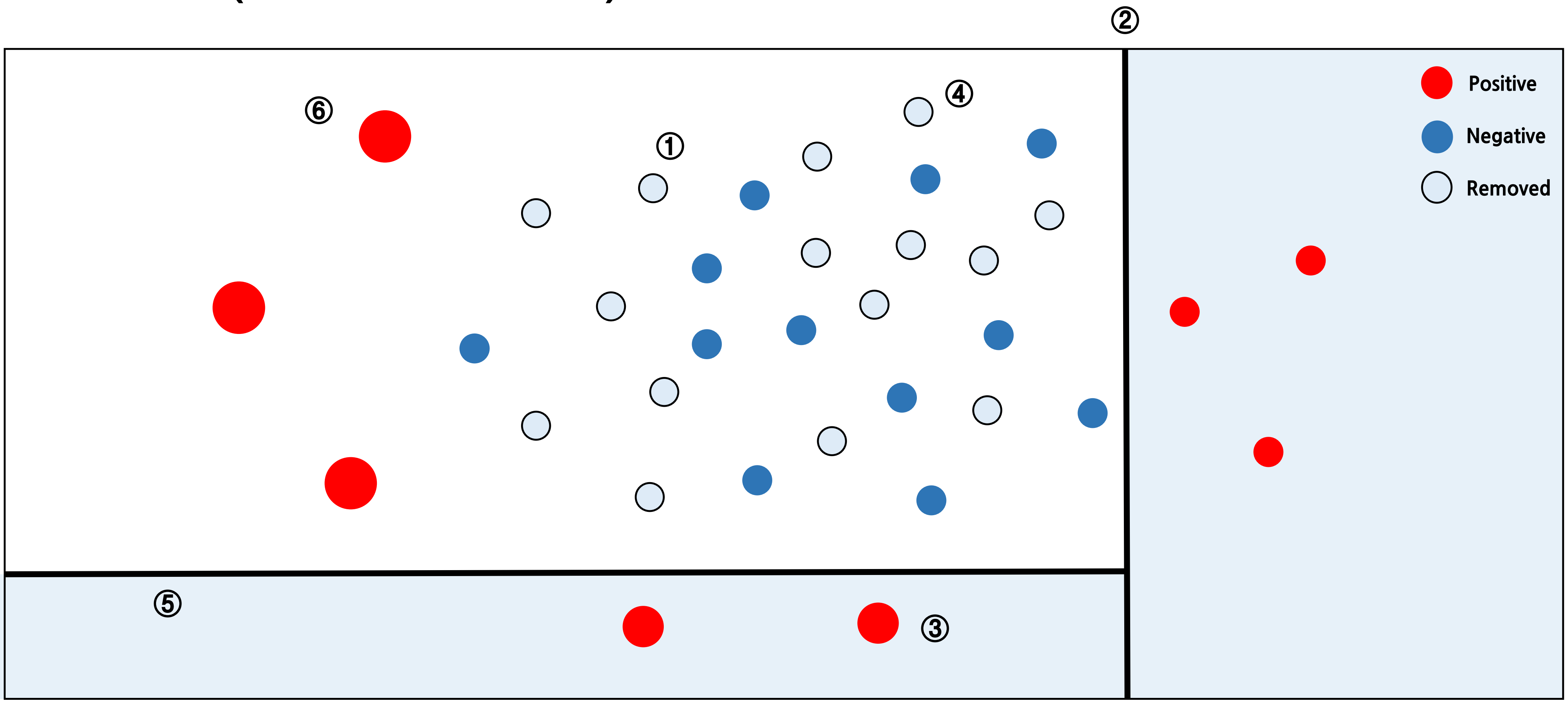
# Ⅲ. RUSBoost vs. SMOTEBoost

❖ RUSBoost(RUS + AdaBoost)



# Ⅲ. RUSBoost vs. SMOTEBoost

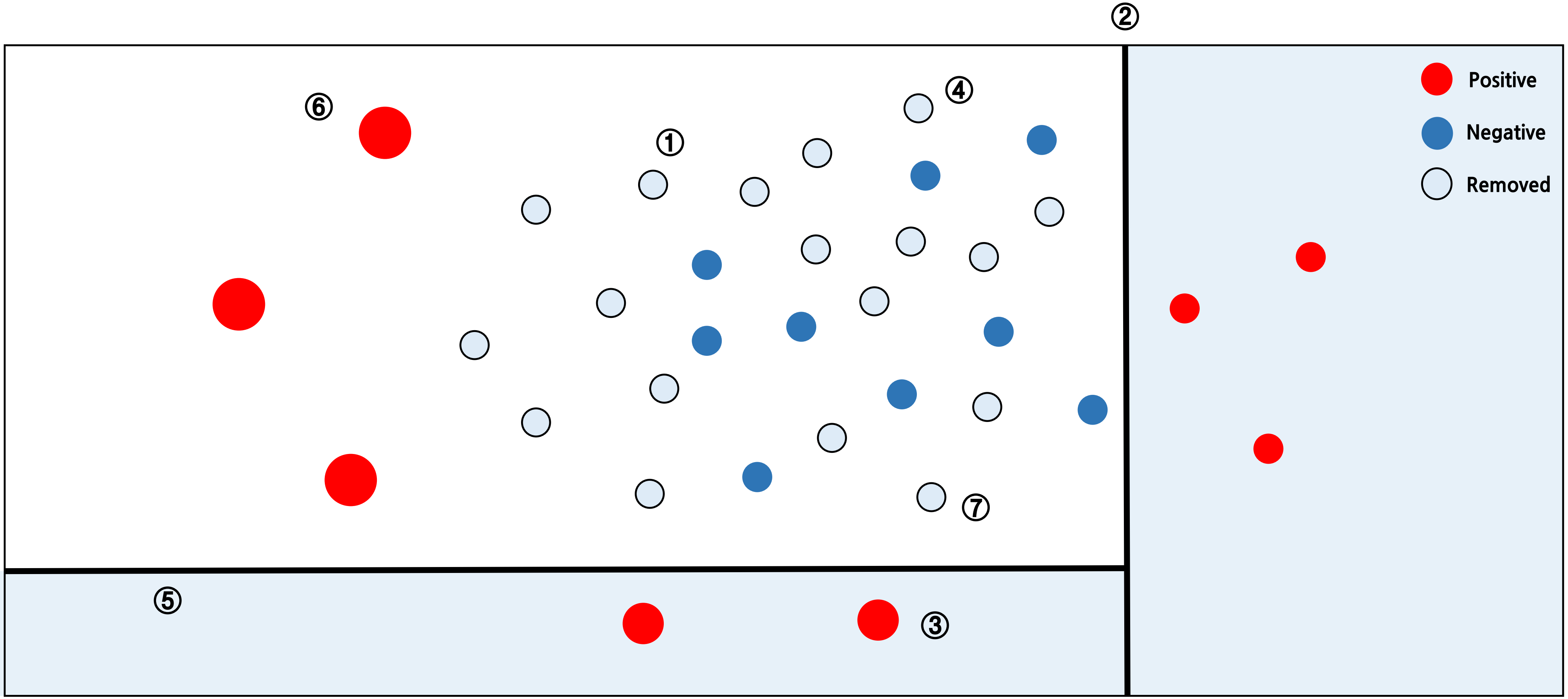
## ❖ RUSBoost(RUS + AdaBoost)




Feature Space

# III. RUSBoost vs. SMOTEBoost

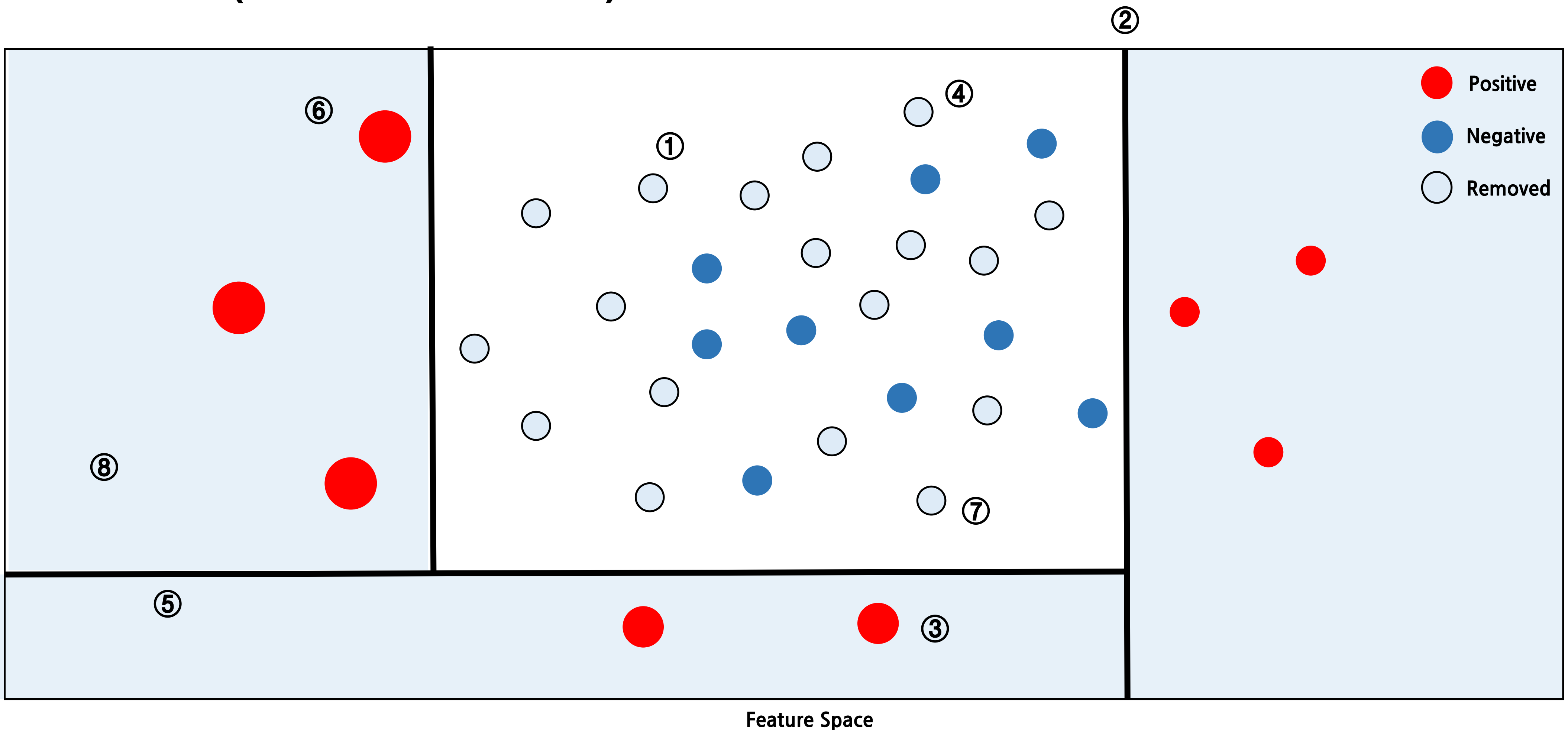
❖ RUSBoost(RUS + AdaBoost)




Feature Space

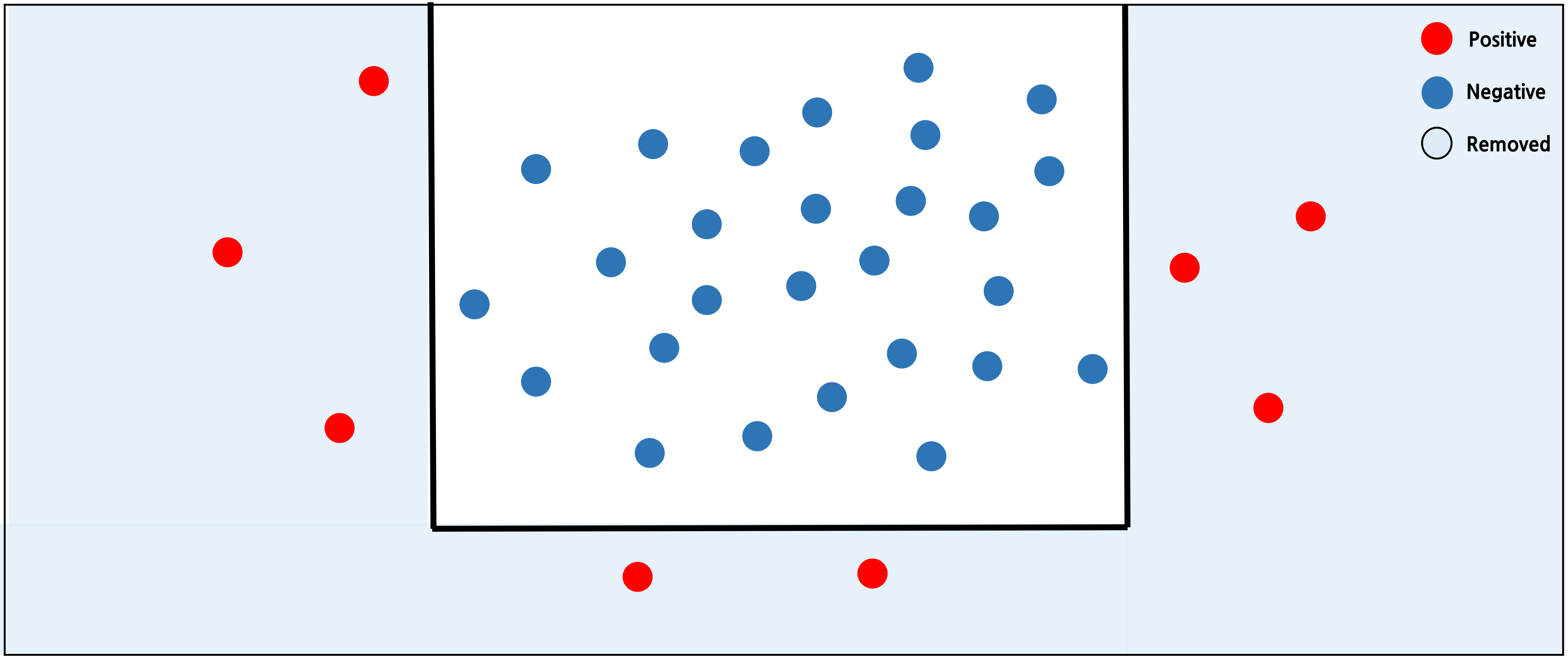
# III. RUSBoost vs. SMOTEBoost

## ❖ RUSBoost(RUS + AdaBoost)



# Ⅲ. RUSBoost vs. SMOTEBoost

❖ RUSBoost(RUS + AdaBoost)



# III. RUSBoost vs. SMOTEBoost

❖ RUSBoost Pesudo code

**Algorithm RUSBoost**

**Given:** Set $S$ of examples $(x_1, y_1), ..., (x_m, y_m)$ with minority class $y^r \in Y$, $|Y| = 2$
Weak learner, $WeakLearn$
Number of iterations, $T$
Desired percentage of total instances to be represented by the minority class, $N$

1 Initialize $D_1(i) = \frac{1}{m}$ for all $i$.
2 Do for $t = 1, 2, ..., T$
  a Create temporary training dataset $S'_t$ with distribution $D'_t$ using random undersampling
  b Call $WeakLearn$, providing it with examples $S'_t$ and their weights $D'_t$.
  c Get back a hypothesis $h_t : X \times Y \rightarrow [0, 1]$.
  d Calculate the pseudo-loss (for $S$ and $D_t$):
  $\epsilon_t = \sum_{(i,y):y_i \neq y} D_t(i)(1 - h_t(x_i, y_i) + h_t(x_i, y))$.
  e Calculate the weight update parameter:
  $\alpha_t = \frac{\epsilon_t}{1 - \epsilon_t}$.
  f Update $D_t$:
  $D_{t+1}(i) = D_t(i)\alpha_t^{\frac{1}{2}(1+h_t(x_i,y_i)-h_t(x_i,y:y\neq y_i))}$.
  g Normalize $D_{t+1}$: Let $Z_t = \sum_i D_{t+1}(i)$.
  $D_{t+1}(i) = \frac{D_{t+1}(i)}{Z_t}$.
3 Output the final hypothesis:

$$H(x) = \underset{y \in Y}{\operatorname{argmax}} \sum_{t=1}^{T} h_t(x, y) \log \frac{1}{\alpha_t}.$$

# III. RUSBoost vs. SMOTEBoost

❖ RUSBoost Pseudo code

**Algorithm RUSBoost**

**Given:** Set $S$ of examples $(x_1, y_1), ..., (x_m, y_m)$ with minority class $y^r \in Y, |Y| = 2$

Weak learner, $WeakLearn$

Number of iterations, $T$

Desired percentage of total instances to be represented by the minority class, $N$

1 Initialize $D_1(i) = \frac{1}{m}$ for all $i$.

2 Do for $t = 1, 2, ..., T$

a Create temporary training dataset $S'_t$ with distribution $D'_t$ using random undersampling

b Call $WeakLearn$, providing it with examples $S'_t$ and their weights $D'_t$.

c Get back a hypothesis $h_t : X \times Y \to [0, 1]$.

d Calculate the pseudo-loss (for $S$ and $D_t$):

$$\epsilon_t = \sum_{(i,y):y_i \neq y} D_t(i)(1 - h_t(x_i, y_i) + h_t(x_i, y)).$$

e Calculate the weight update parameter:

$$\alpha_t = \frac{\epsilon_t}{1 - \epsilon_t}.$$

f Update $D_t$:

$$D_{t+1}(i) = D_t(i)\alpha_t^{\frac{1}{2}(1+h_t(x_i,y_i)-h_t(x_i,y:y\neq y_i))}.$$

g Normalize $D_{t+1}$: Let $Z_t = \sum_i D_{t+1}(i)$.

$$D_{t+1}(i) = \frac{D_{t+1}(i)}{Z_t}.$$

3 Output the final hypothesis:

$$H(x) = \underset{y \in Y}{\operatorname{argmax}} \sum_{t=1}^{T} h_t(x, y) \log \frac{1}{\alpha_t}.$$

# III. RUSBoost vs. SMOTEBoost

## ❖ SMOTEBoost : Improving Prediction of the Minority Class in Boosting

- European Conference on Principles of Data Mining and Knowledge Discovery - 2003

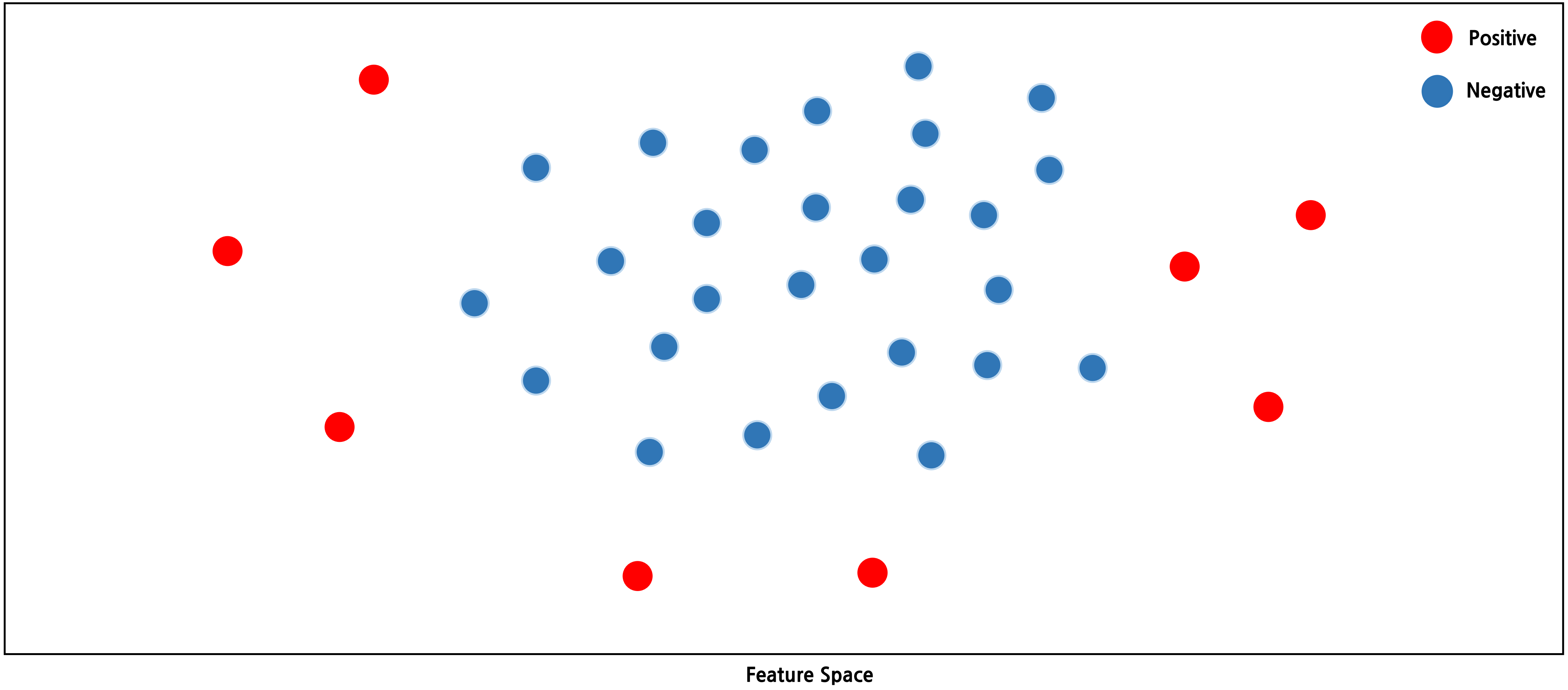
SMOTEBoost: Improving prediction of the minority class in boosting

NV Chawla, A Lazarevic, LO Hall… - European conference on …, 2003 - Springer

Many real world data mining applications involve learning from imbalanced data sets. Learning from data sets that contain very few instances of the minority (or interesting) class usually produces biased classifiers that have a higher predictive accuracy over the majority class (es), but poorer predictive accuracy over the minority class. SMOTE (Synthetic Minority Over-sampling TEchnique) is specifically designed for learning from imbalanced data sets. This paper presents a novel approach for learning from imbalanced data sets, based on a …

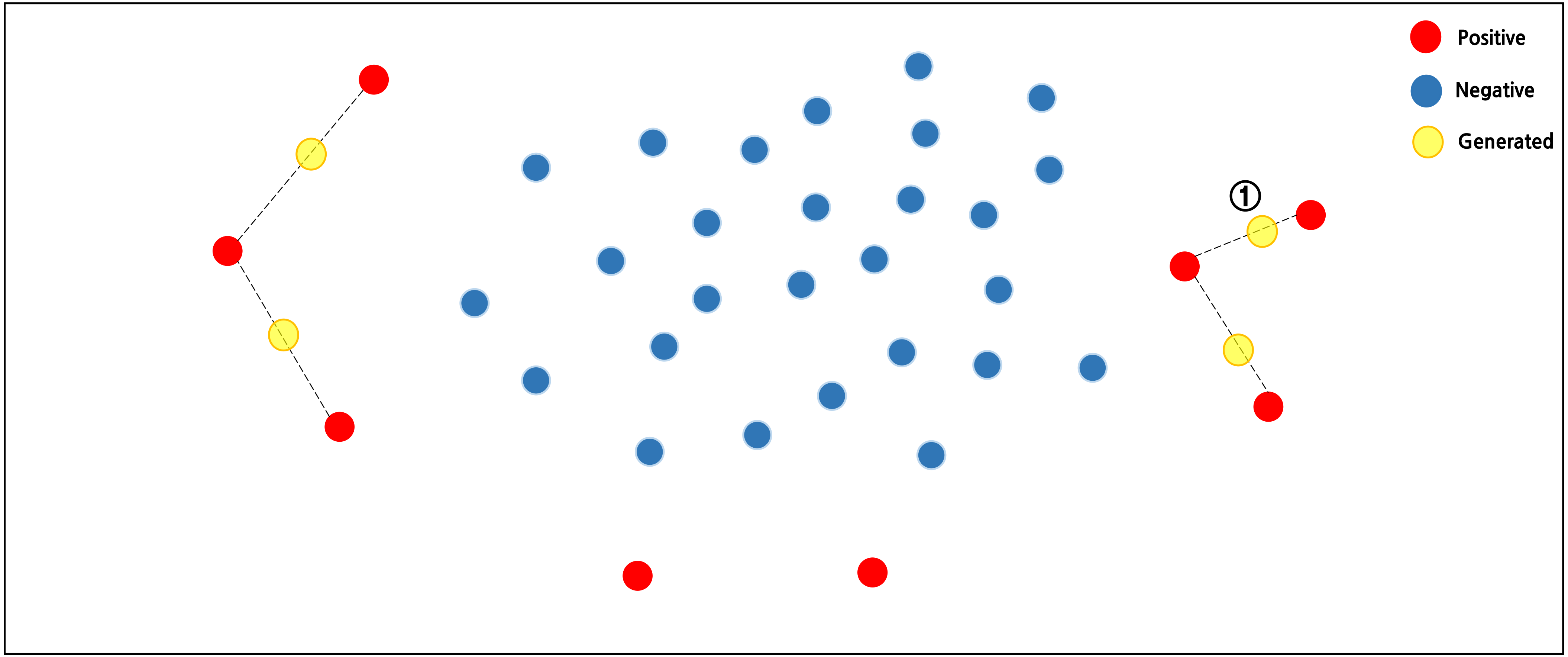
# III. RUSBoost vs. SMOTEBoost

❖ SMOTEBoost(SMOTE + AdaBoost)



# III. RUSBoost vs. SMOTEBoost

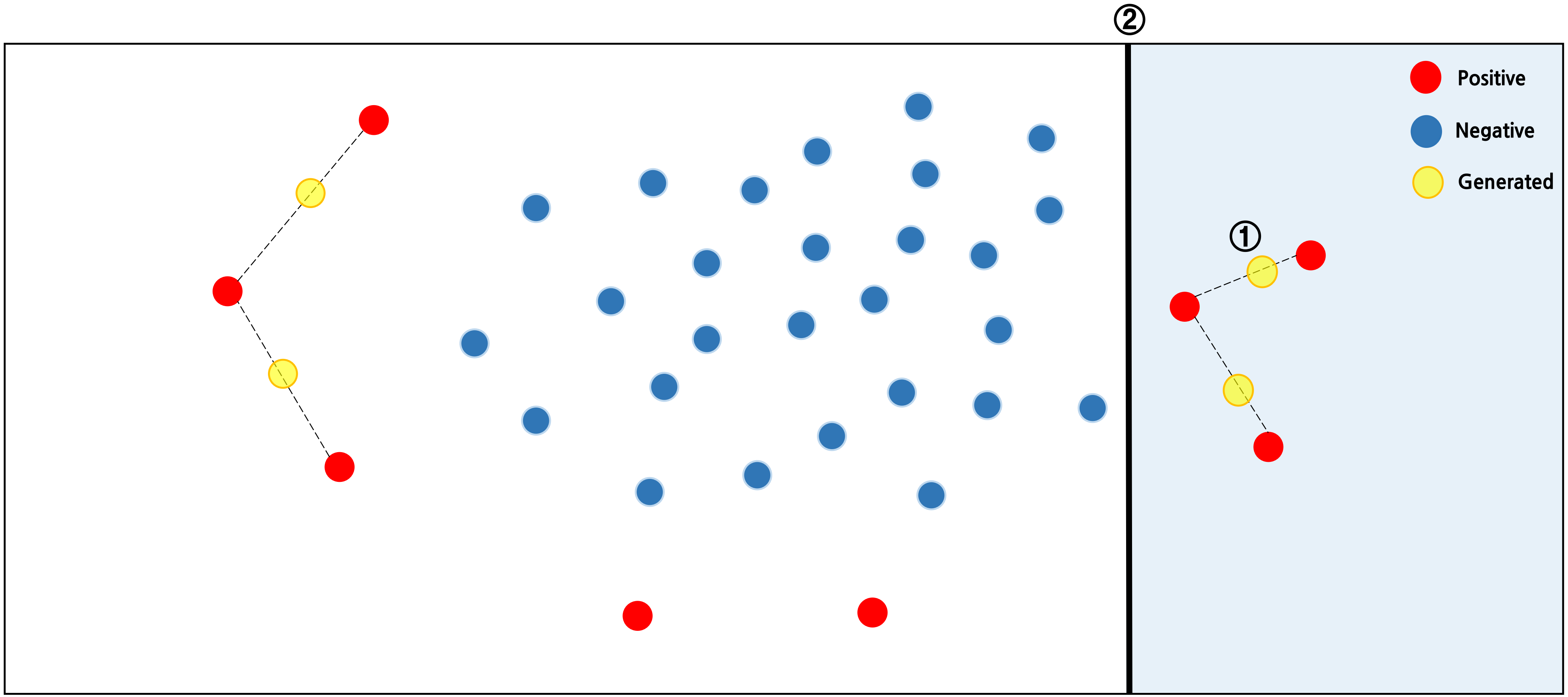
❖ SMOTEBoost(SMOTE + AdaBoost)




Feature Space

# III. RUSBoost vs. SMOTEBoost

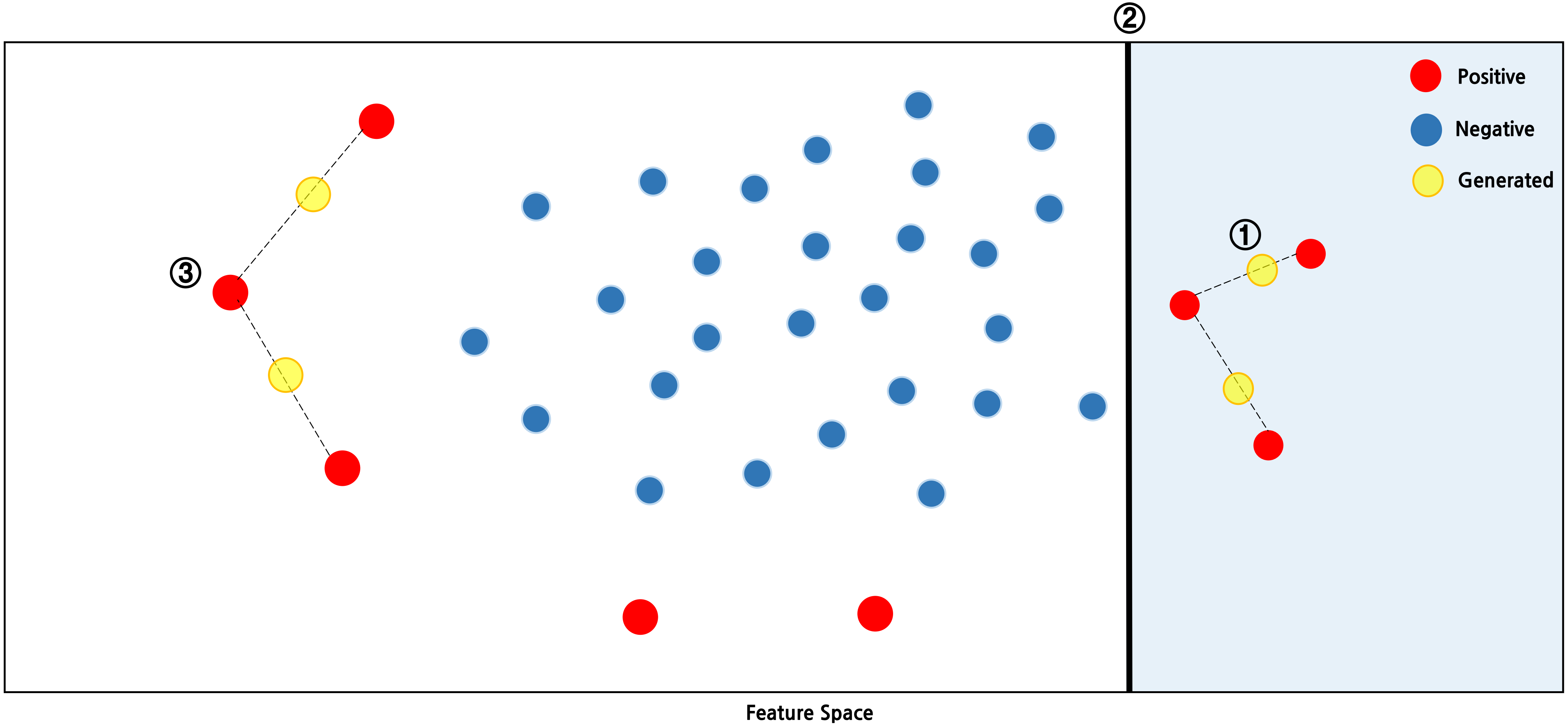
❖ SMOTEBoost(SMOTE + AdaBoost)




Feature Space

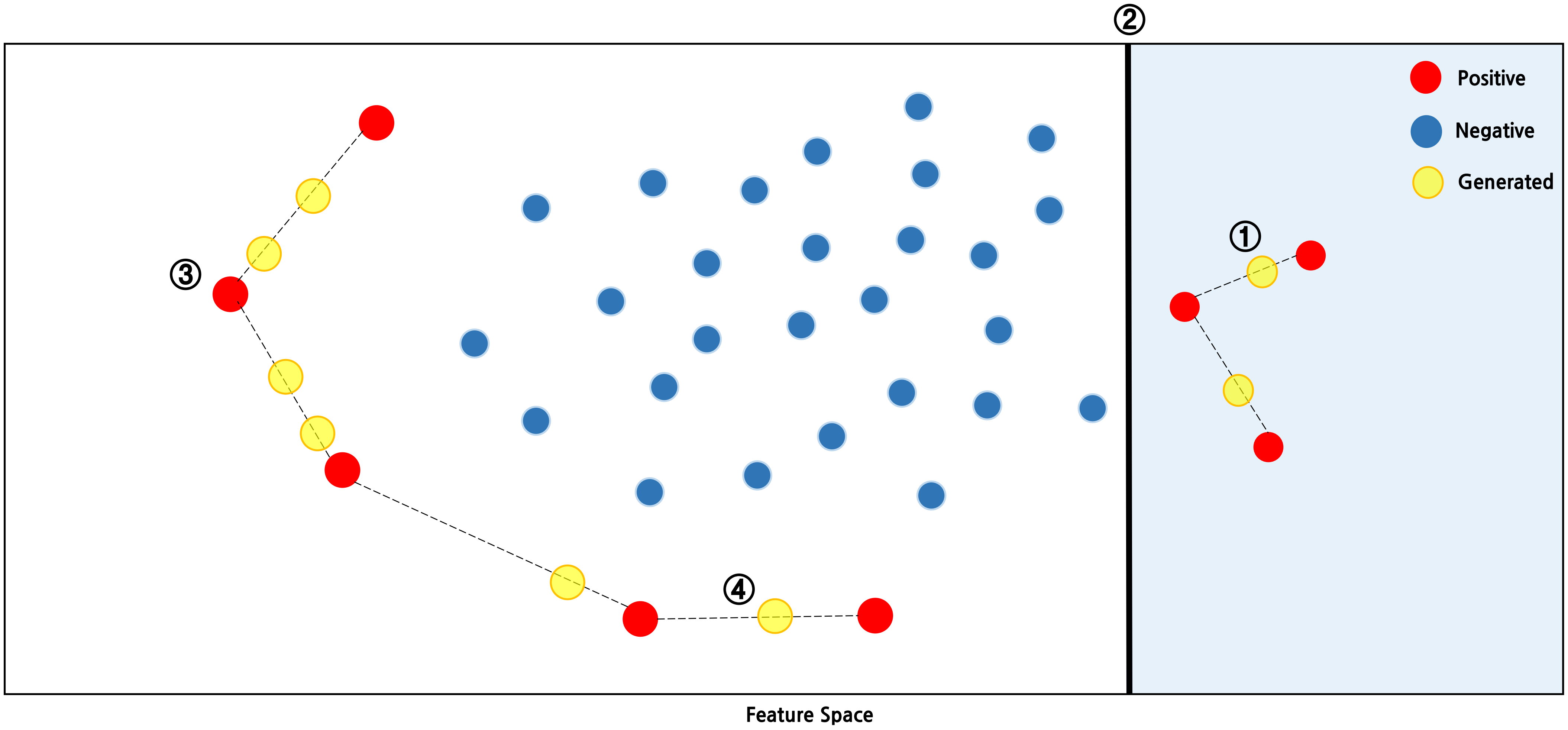
# III. RUSBoost vs. SMOTEBoost

❖ SMOTEBoost(SMOTE + AdaBoost)



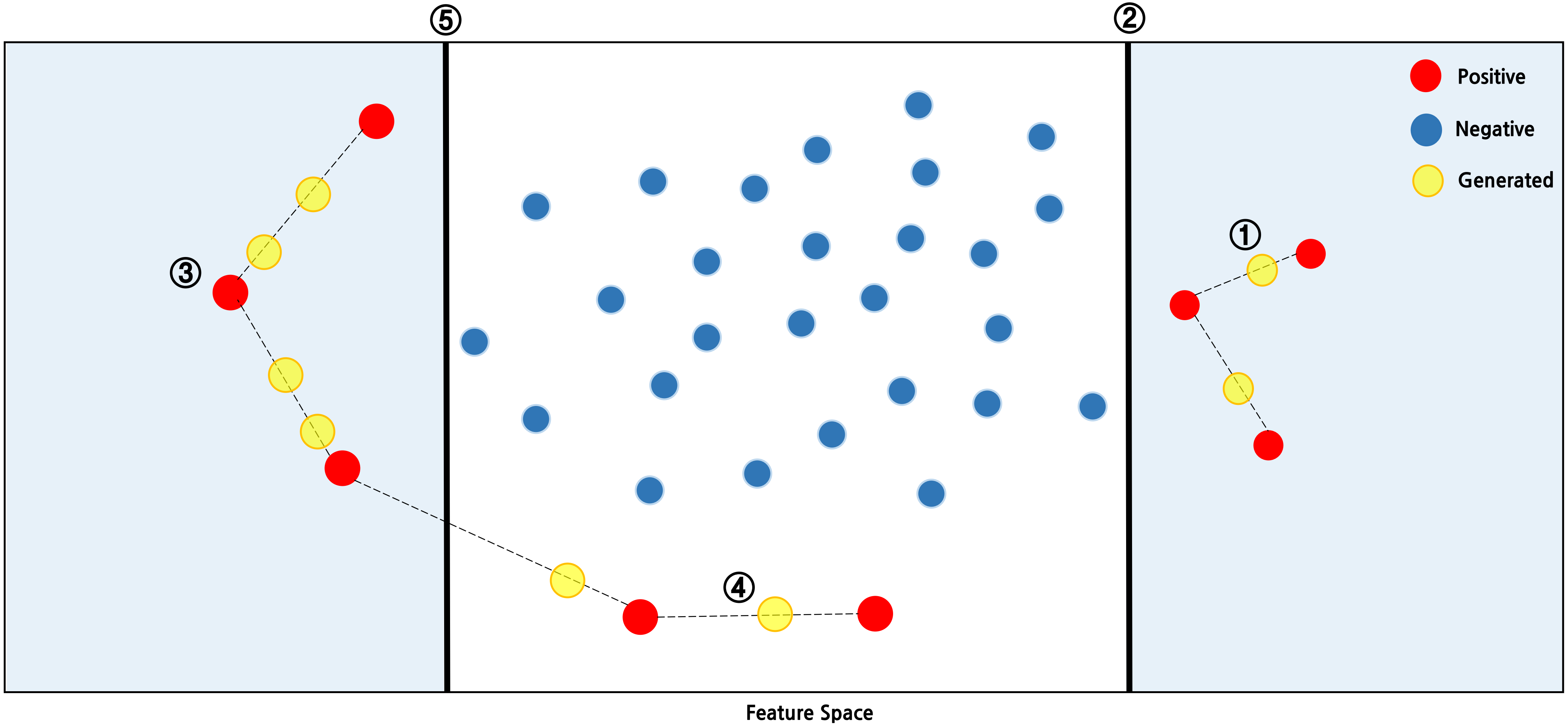
# Ⅲ. RUSBoost vs. SMOTEBoost

❖ SMOTEBoost(SMOTE + AdaBoost)



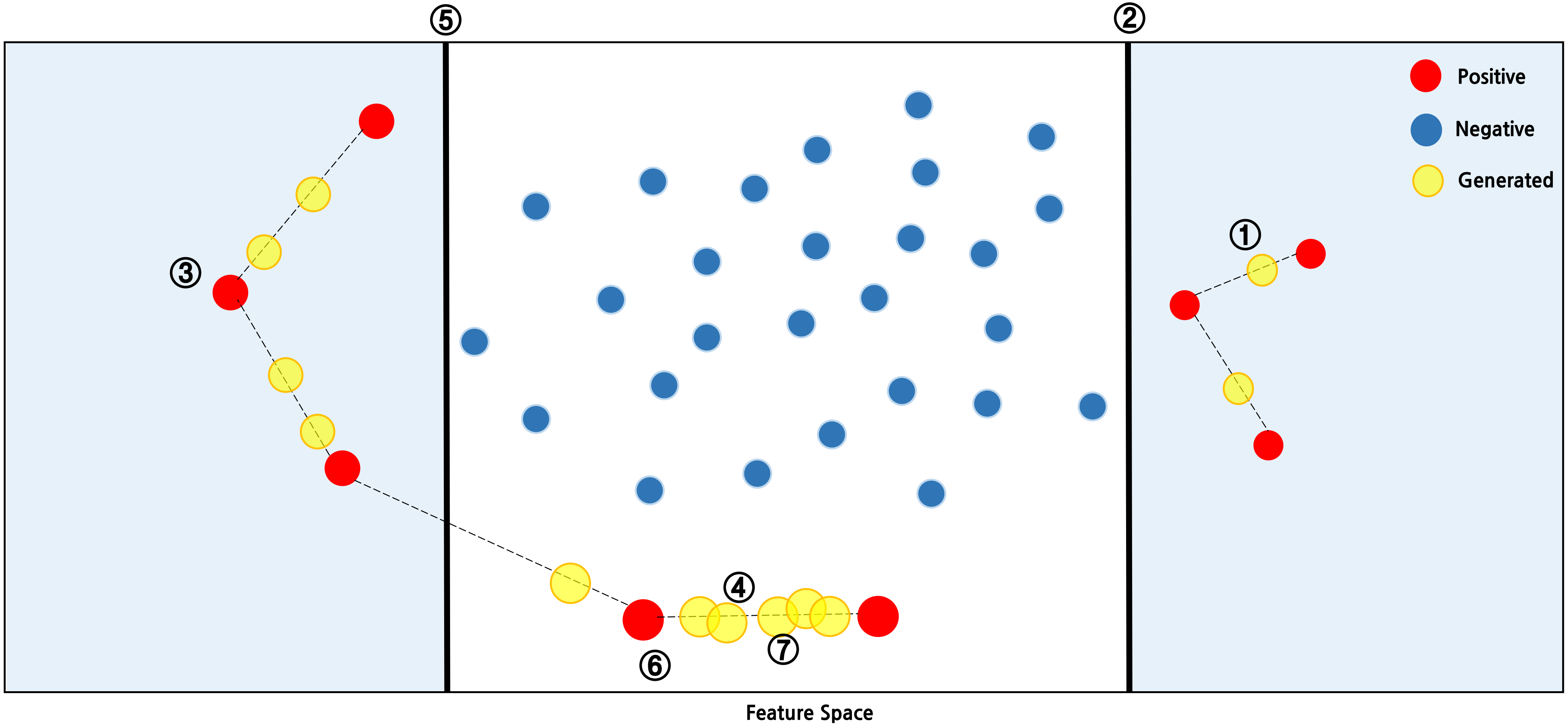
# III. RUSBoost vs. SMOTEBoost

❖ SMOTEBoost(SMOTE + AdaBoost)



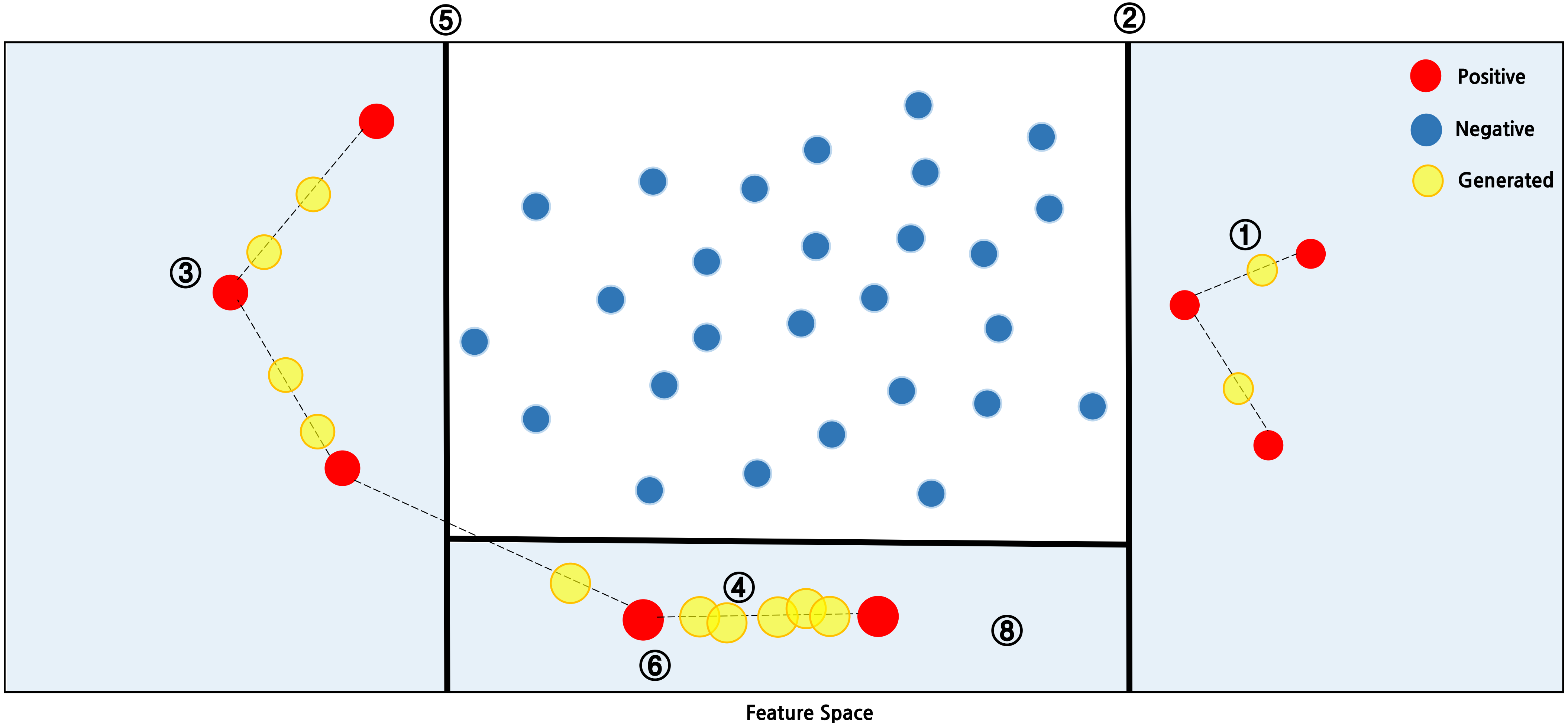
# III. RUSBoost vs. SMOTEBoost

❖ SMOTEBoost(SMOTE + AdaBoost)



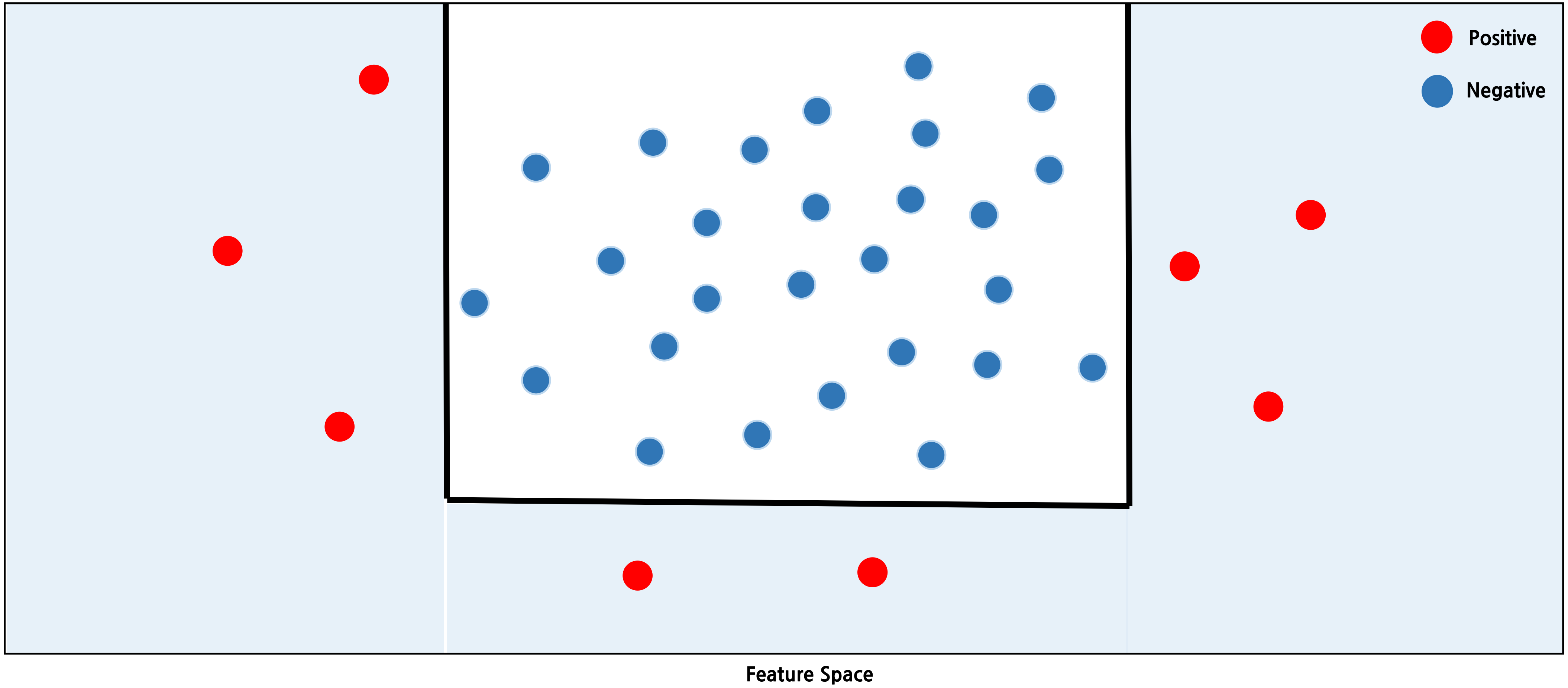
# III. RUSBoost vs. SMOTEBoost

❖ SMOTEBoost(SMOTE + AdaBoost)



# Ⅲ. RUSBoost vs. SMOTEBoost

❖ SMOTEBoost(SMOTE + AdaBoost)



# III. RUSBoost vs. SMOTEBoost

## ❖ SMOTEBoost Pseudo code

- Given: Set S $\{(x_1, y_1), \ldots, (x_m, y_m)\}$ $x_i \in X$, with labels $y_i \in Y = \{1, \ldots, C\}$, where $C_m$, $(C_m < C)$ corresponds to a minority class.
- Let $B = \{(i, y): i = 1,\ldots,m, y \neq y_i\}$
- Initialize the distribution $D_1$ over the examples, such that $D_1(i) = 1/m$.
- For $t = 1, 2, 3, 4, \ldots T$
  1. Modify distribution $D_t$ by creating $N$ synthetic examples from minority class $C_m$ using the SMOTE algorithm
  2. Train a weak learner using distribution $D_t$
  3. Compute weak hypothesis $h_t$: $X \times Y \to [0, 1]$
  4. Compute the pseudo-loss of hypothesis $h_t$:
     $$\varepsilon_t = \sum_{(i,y)\in B} D_t(i,y)(1 - h_t(x_i, y_i) + h_t(x_i, y))$$
  5. Set $\beta_t = \varepsilon_t / (1 - \varepsilon_t)$ and $w_t = (1/2)\cdot(1-h_t(x_i,y)+h_t(x_i,y_i))$
  6. Update $D_t$ : $D_{t+1}(i, y) = (D_t(i,y)/Z_t)\cdot \beta_t^{w_t}$
     where $Z_t$ is a normalization constant chosen such that $D_{t+1}$ is a distribution.
- Output the final hypothesis: $h_{fn} = \arg\max_{y\in Y} \sum_{t=1}^{T} (\log \frac{1}{\beta_t}) \cdot h_t(x, y)$

# III. RUSBoost vs. SMOTEBoost

## ❖ SMOTEBoost Pseudo code

- Given: Set S $\{(x_1, y_1), \ldots, (x_m, y_m)\}$ $x_i \in X$, with labels $y_i \in Y = \{1, \ldots, C\}$, where $C_m$, ($C_m < C$) corresponds to a minority class.
- Let $B = \{(i, y): i = 1,\ldots,m, y \neq y_i\}$
- Initialize the distribution $D_1$ over the examples, such that $D_1(i) = 1/m$.
- For $t = 1, 2, 3, 4, \ldots T$
  1. Modify distribution $D_t$ by creating $N$ synthetic examples from minority class $C_m$ using the SMOTE algorithm
  2. Train a weak learner using distribution $D_t$
  3. Compute weak hypothesis $h_t$: $X \times Y \rightarrow [0, 1]$
  4. Compute the pseudo-loss of hypothesis $h_t$:
     $$\varepsilon_t = \sum_{(i,y)\in B} D_t(i,y)(1 - h_t(x_i, y_i) + h_t(x_i, y))$$
  5. Set $\beta_t = \varepsilon_t / (1 - \varepsilon_t)$ and $w_t = (1/2)\cdot(1-h_t(x_i,y)+h_t(x_i,y_i))$
  6. Update $D_t$: $D_{t+1}(i, y) = (D_t(i,y) / Z_t) \cdot \beta_t^{w_t}$
     where $Z_t$ is a normalization constant chosen such that $D_{t+1}$ is a distribution.
- Output the final hypothesis: $h_{fn} = \arg\max_{y\in Y} \sum_{t=1}^{T} (\log \frac{1}{\beta_t}) \cdot h_t(x, y)$

# Contents

# IV. Result of experiments

## ❖ Datasets

- Experiments were conducted on 15 class imbalance data.
- To ensure independence of each result value, 10-fold cross validation was performed 10 times in total.

TABLE I
DATA SET CHARACTERISTICS

| Dataset | Size | # min | % min | # attr |
|---|---|---|---|---|
| SP3 | 3541 | 47 | 1.33 | 43 |
| MAMMOGRAPHY | 11183 | 260 | 2.32 | 7 |
| SOLARFLAREF | 1389 | 51 | 3.67 | 13 |
| CAR3 | 1728 | 69 | 3.99 | 7 |
| CCCS12 | 282 | 16 | 5.67 | 9 |
| SP1 | 3649 | 229 | 6.28 | 43 |
| PC1 | 1107 | 76 | 6.87 | 16 |
| GLASS3 | 214 | 17 | 7.94 | 10 |
| CM1 | 505 | 48 | 9.50 | 16 |
| PENDIGITS5 | 10992 | 1055 | 9.60 | 17 |
| SATIMAGE4 | 6435 | 626 | 9.73 | 37 |
| ECOLI4 | 336 | 35 | 10.42 | 8 |
| SEGMENT5 | 2310 | 330 | 14.29 | 20 |
| CONTRA2 | 1473 | 333 | 22.61 | 10 |
| VEHICLE1 | 846 | 212 | 25.06 | 19 |

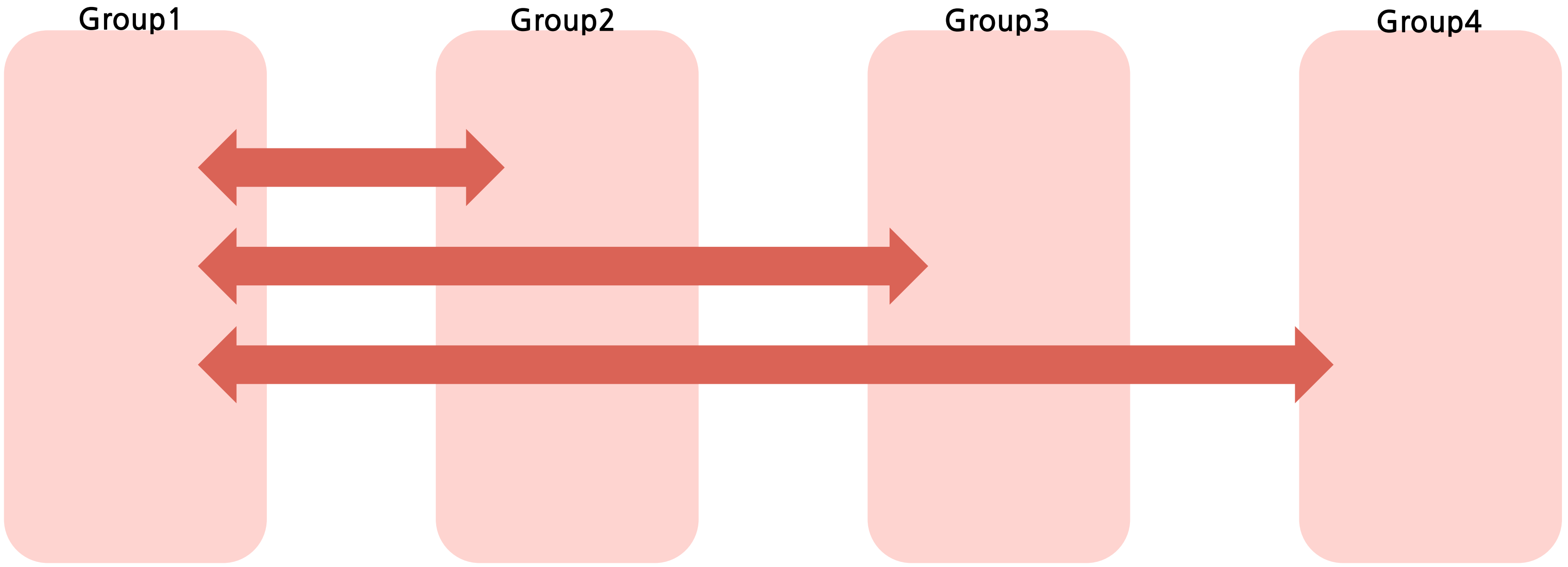
## ❖ Multiple Comparison(다중 비교)

- After the analysis of variance , this hypothesis test is conducted when the null hypothesis is rejected.

# Ⅳ. Result of experiments

## ❖ Multiple Comparison(다중 비교)

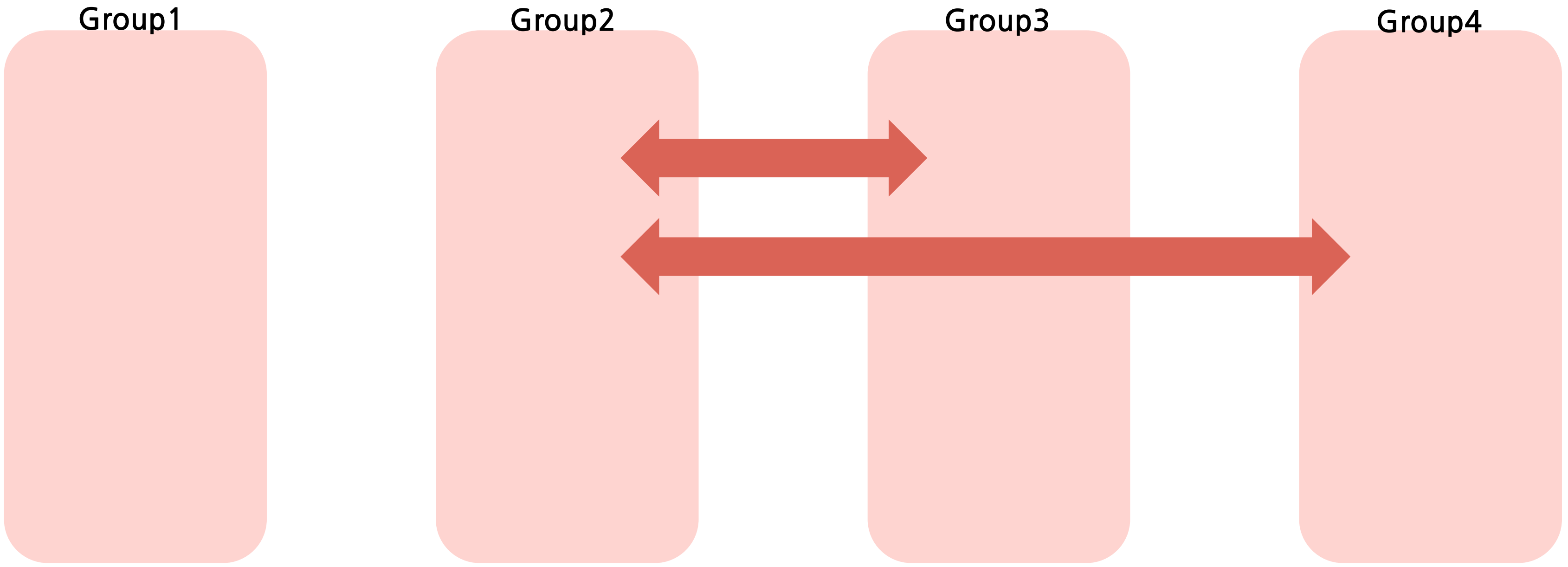
- After the analysis of variance , this hypothesis test is conducted when the null hypothesis is rejected.
- Tests are conducted by grouping the two groups together and testing how similar the groups are.



# Ⅳ. Result of experiments

## ❖ Multiple Comparison(다중 비교)

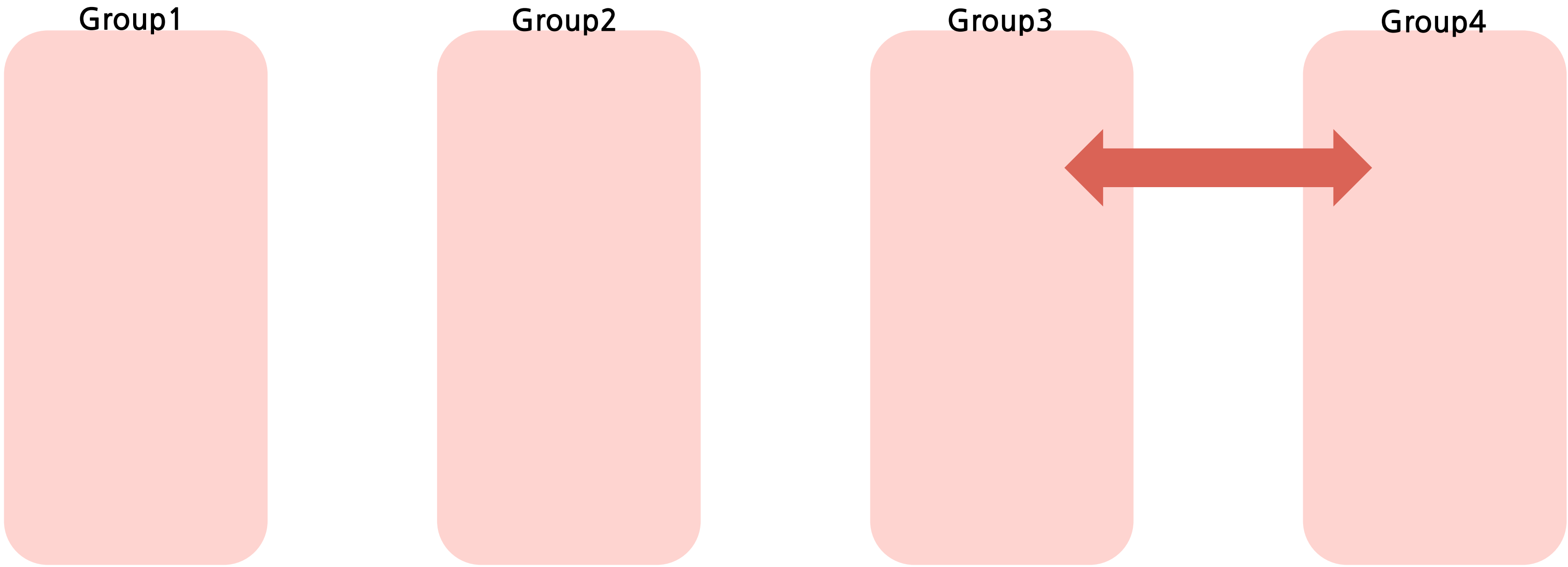
- After the analysis of variance , this hypothesis test is conducted when the null hypothesis is rejected.
- Tests are conducted by grouping the two groups together and testing how similar the groups are.



# Ⅳ. Result of experiments

## ❖ Multiple Comparison(다중 비교)

- After the analysis of variance , this hypothesis test is conducted when the null hypothesis is rejected.
- Tests are conducted by grouping the two groups together and testing how similar the groups are.



# Ⅳ. Result of experiments

❖ Result

TABLE II
AVERAGE PERFORMANCES OF THE SAMPLING TECHNIQUES ACROSS ALL LEARNERS AND DATA SETS

| Technique | A-ROC | | K-S | | A-PRC | | F-measure | |
|---|---|---|---|---|---|---|---|---|
| | Mean | HSD | Mean | HSD | Mean | HSD | Mean | HSD |
| RUSBoost | 0.8704 | A | 0.7325 | A | 0.5629 | A | 0.4971 | A |
| SMOTEBoost | 0.8674 | A | 0.7284 | A | 0.5707 | A | 0.4976 | A |
| AdaBoost | 0.8394 | B | 0.6813 | B | 0.5253 | B | 0.4506 | C |
| RUS | 0.8243 | C | 0.6507 | D | 0.3916 | E | 0.4228 | D |
| SMOTE | 0.8199 | C | 0.6633 | C | 0.4776 | C | 0.4755 | B |
| None | 0.7670 | D | 0.5355 | E | 0.4308 | D | 0.4117 | E |

# Contents



DMQA

# V. Conclusion

## ❖ Conclusion

- You should use the appropriate algorithm for your problem situation.

- For example, if you do not know whether the data is very large and can be operated on the memory, it is recommended to select the RUS algorithms.

- If the training dataset is very small and the number of positive (minority) class is also small, you should use the SMOTE algorithms.

# Thank you.