# Imbalanced Data

Nguyen Khoa



Year 2024

AI VIET NAM
@aivietnam.edu.vn

# CONTENT

# Data Pre-processing

## Introduction

Advantages:

- Can be applied to any existing learning tool
- The chosen models are biased to the goals of the user (because the data distribution was previously changed to match these goals), and thus it is expected that the models are more interpretable in terms of these goals.

Disadvantages:

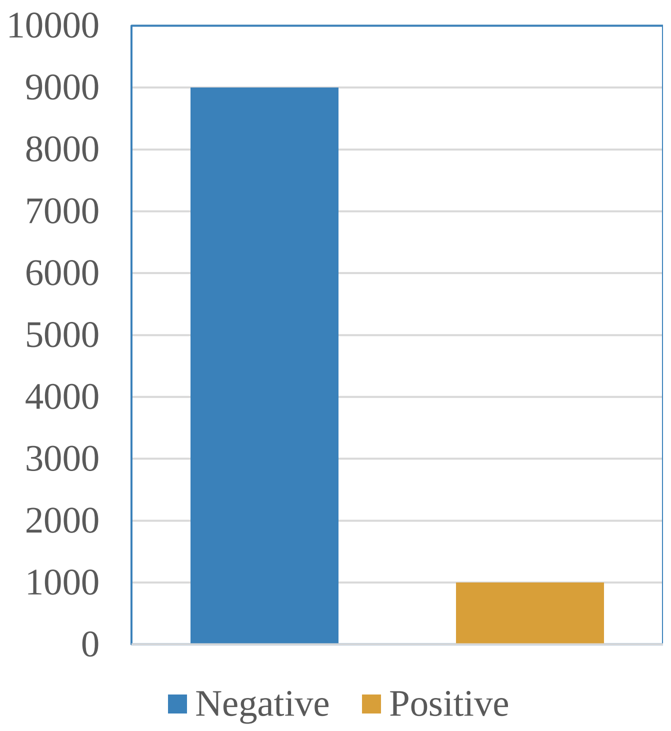
- Mapping the given data distribution into an optimal new distribution according to the user goals is not easy.
- It was proved for classification tasks that a perfectly balanced distribution does not always provide optimal results

AI VIET NAM
@aivietnam.edu.vn

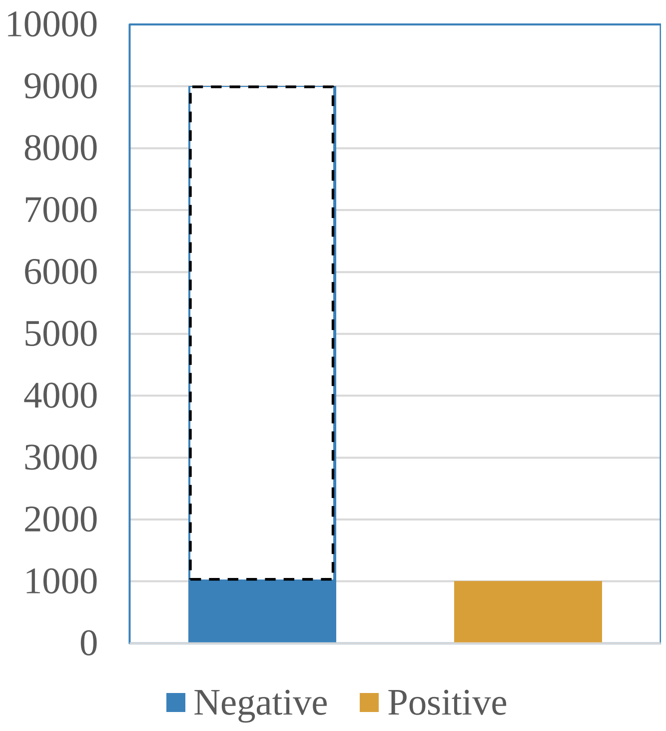
# Data Pre-processing

## Re-Sampling / Random Under-sampling and Over-sampling

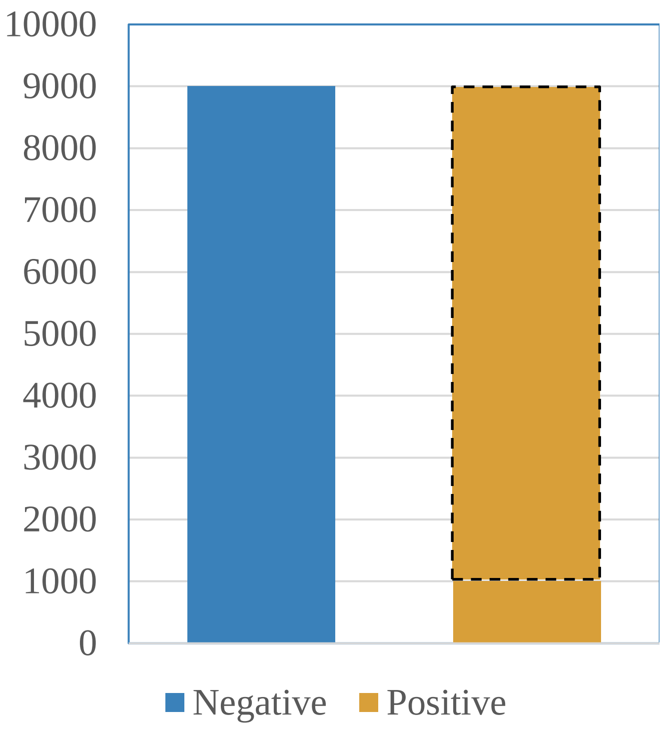
### Original Data




### Undersampling Data




### Oversampling Data

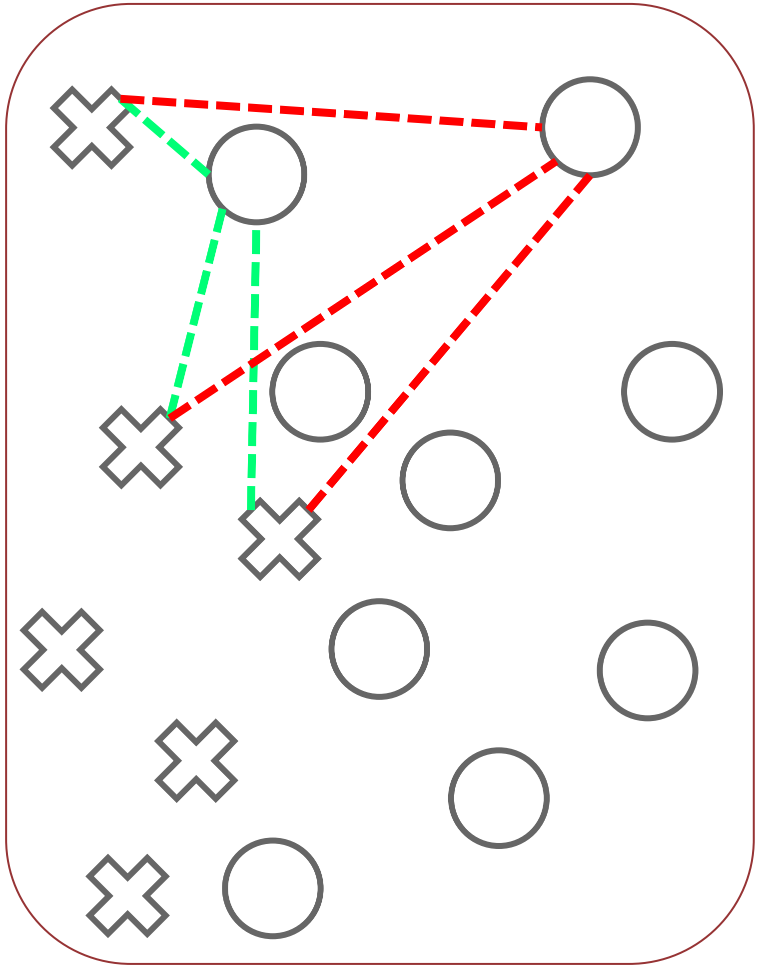


AI VIET NAM
@aivietnam.edu.vn

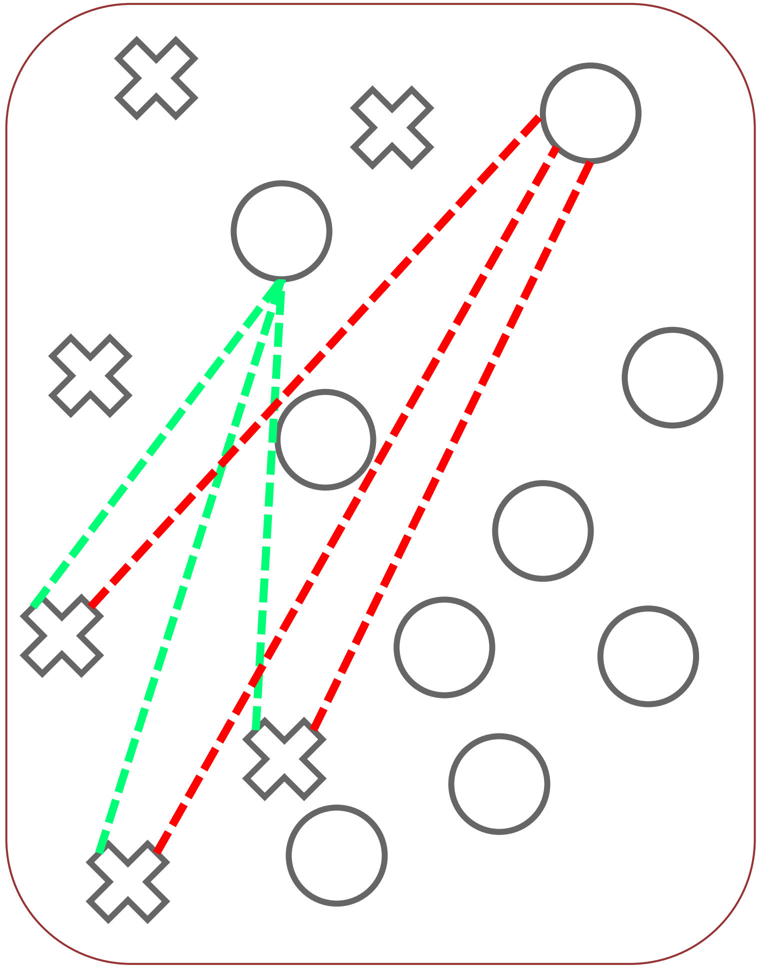
# Data Pre-processing

## Re-Sampling / Distance-Based / NearMiss

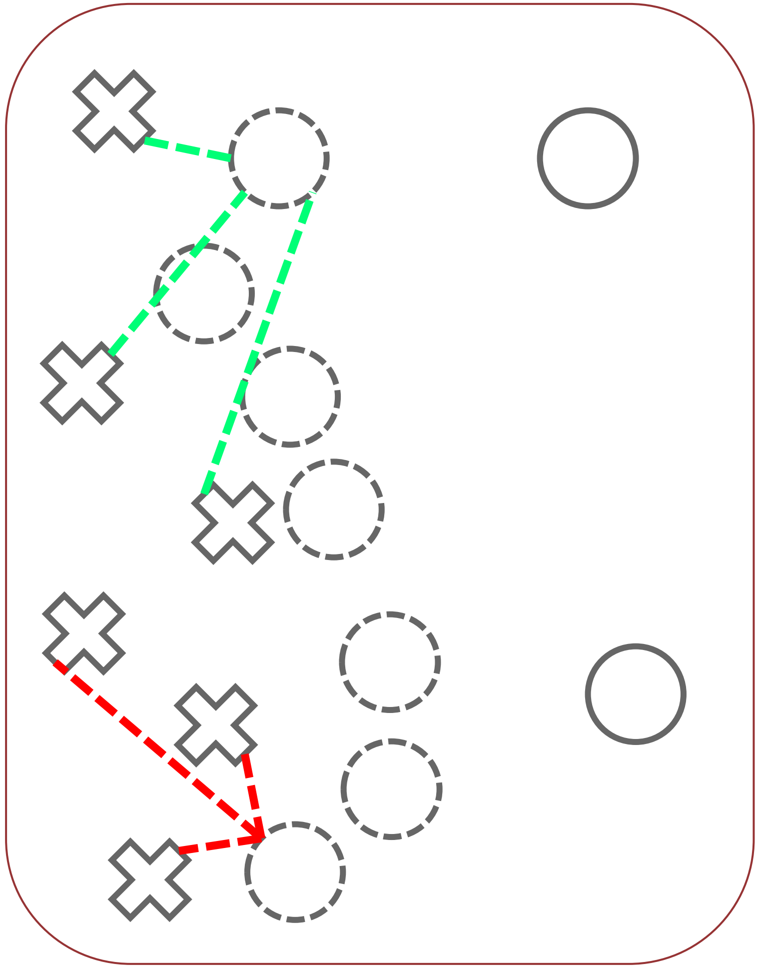
NearMiss1



NearMiss2



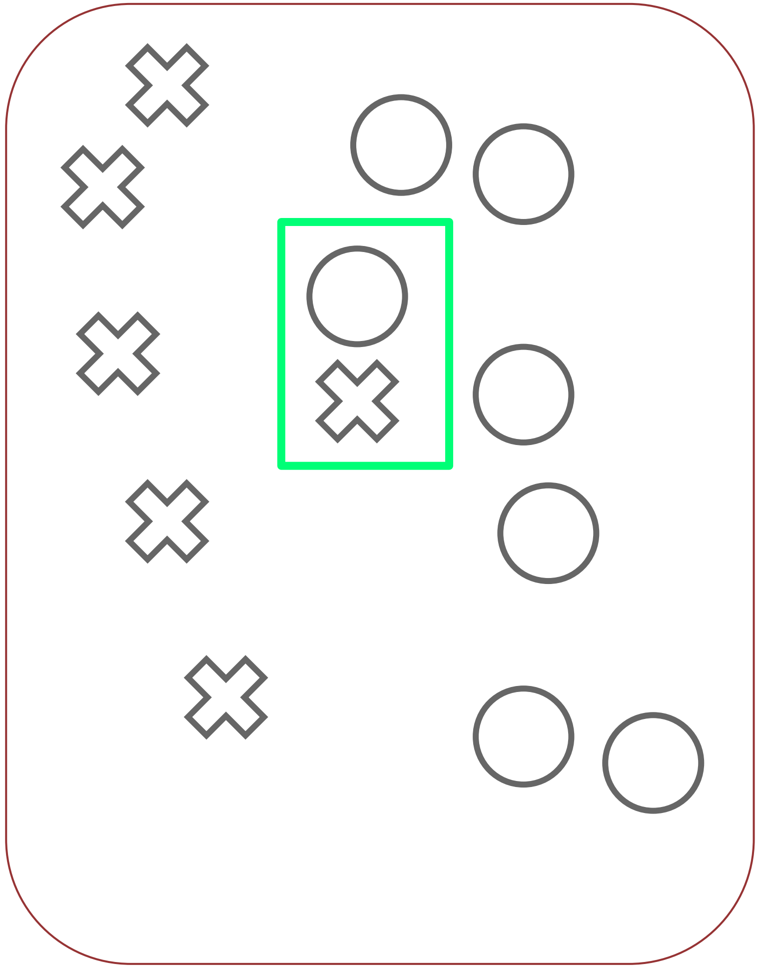
NearMiss1



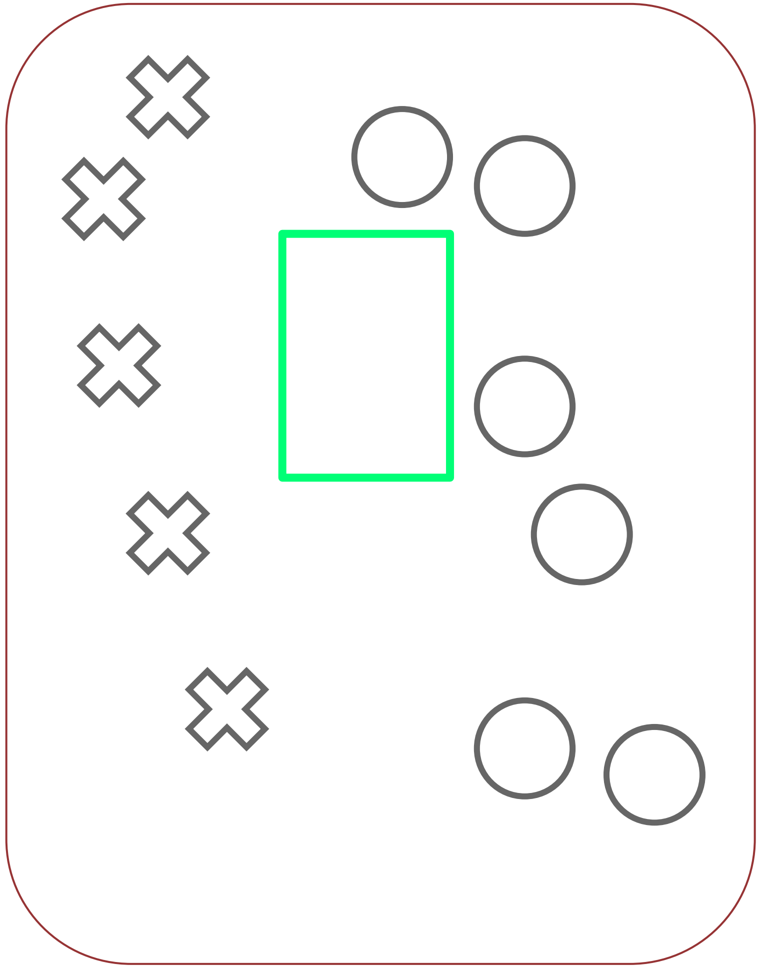
# Data Pre-processing

## Re-Sampling / Data Cleaning / Tomek Links

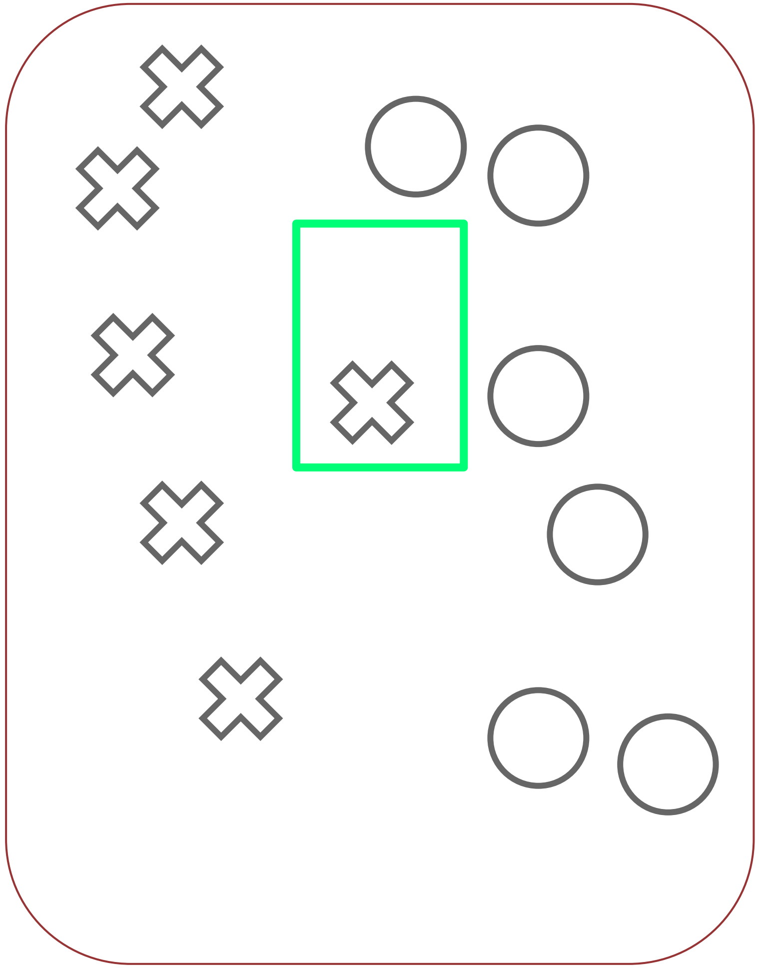
**Tomek Links**



**Approach 1: Remove All**

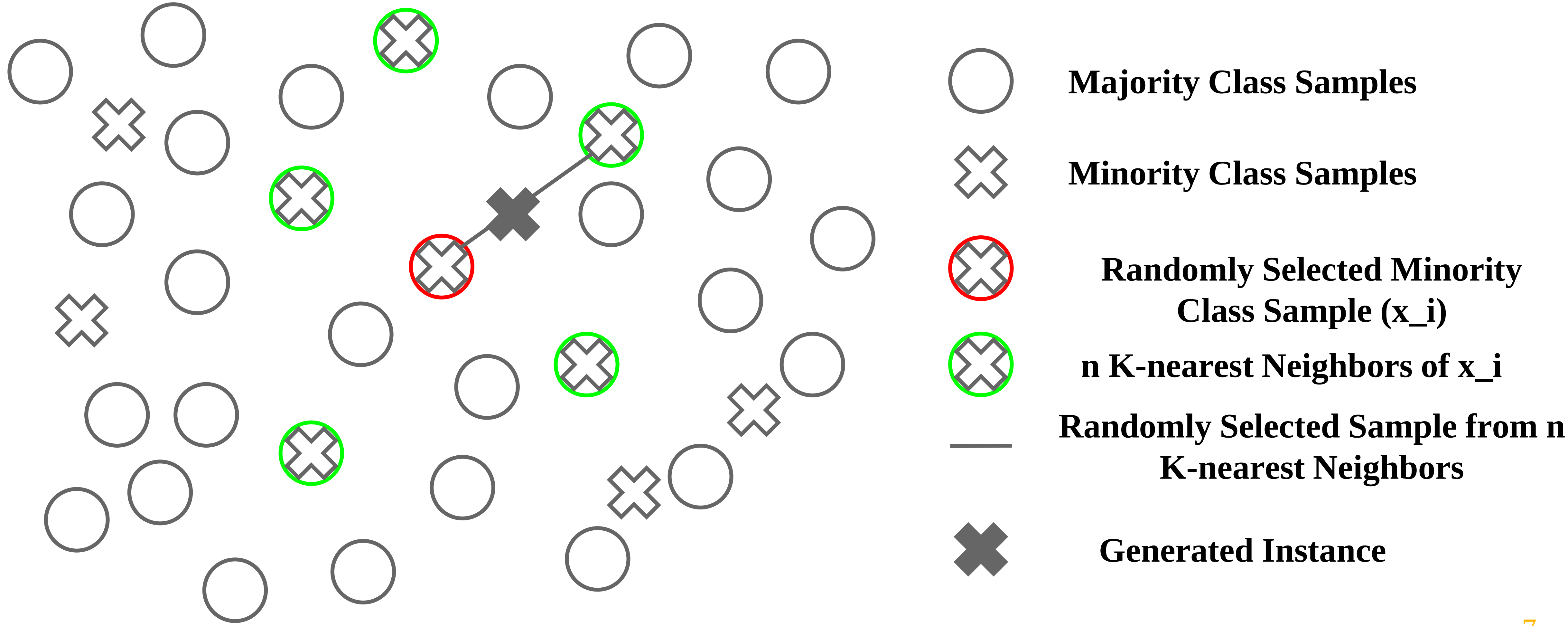


**Approach 1: Remove Majority**



# Data Pre-processing

## Re-Sampling / Synthesising New Data / SMOTE



# Special-purpose Learning Methods

## Introduction

Advantages:

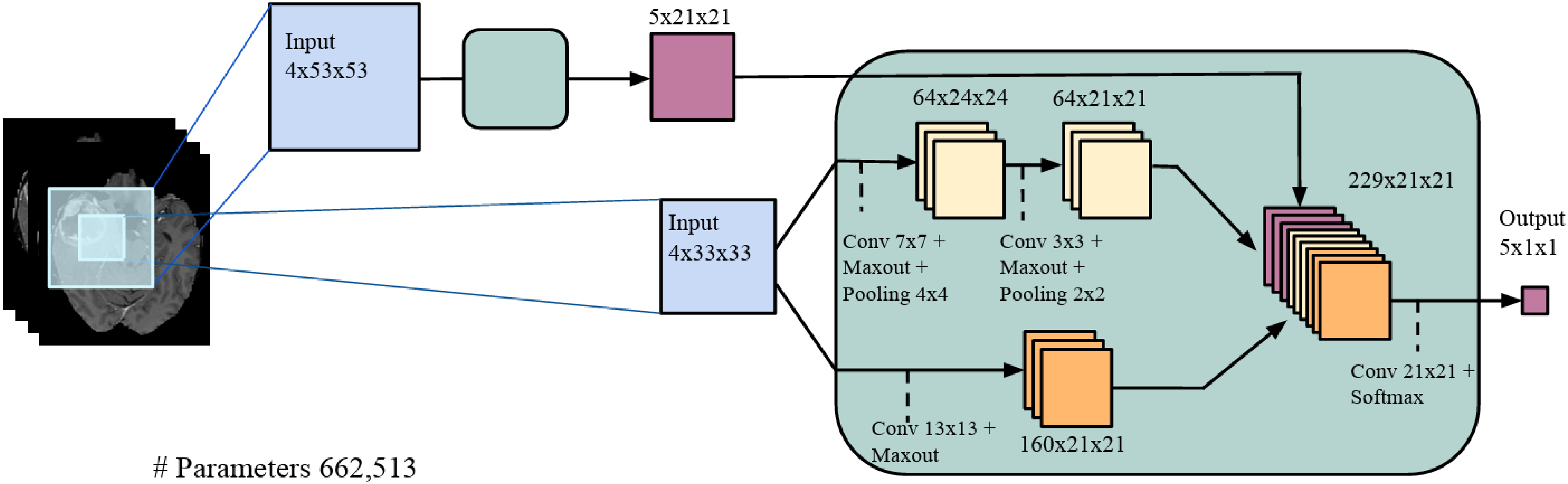
- The user goals are incorporated directly into the models

Disadvantages:

- restricted in his choice to the learning algorithms that have been modified to be able to optimise his goals, or has to develop new algorithms for the task
- may be necessary to introduce further modifications in the algorithm which may not be straightforward
- requires a deep knowledge of the learning algorithms implementations

# Special-purpose Learning Methods

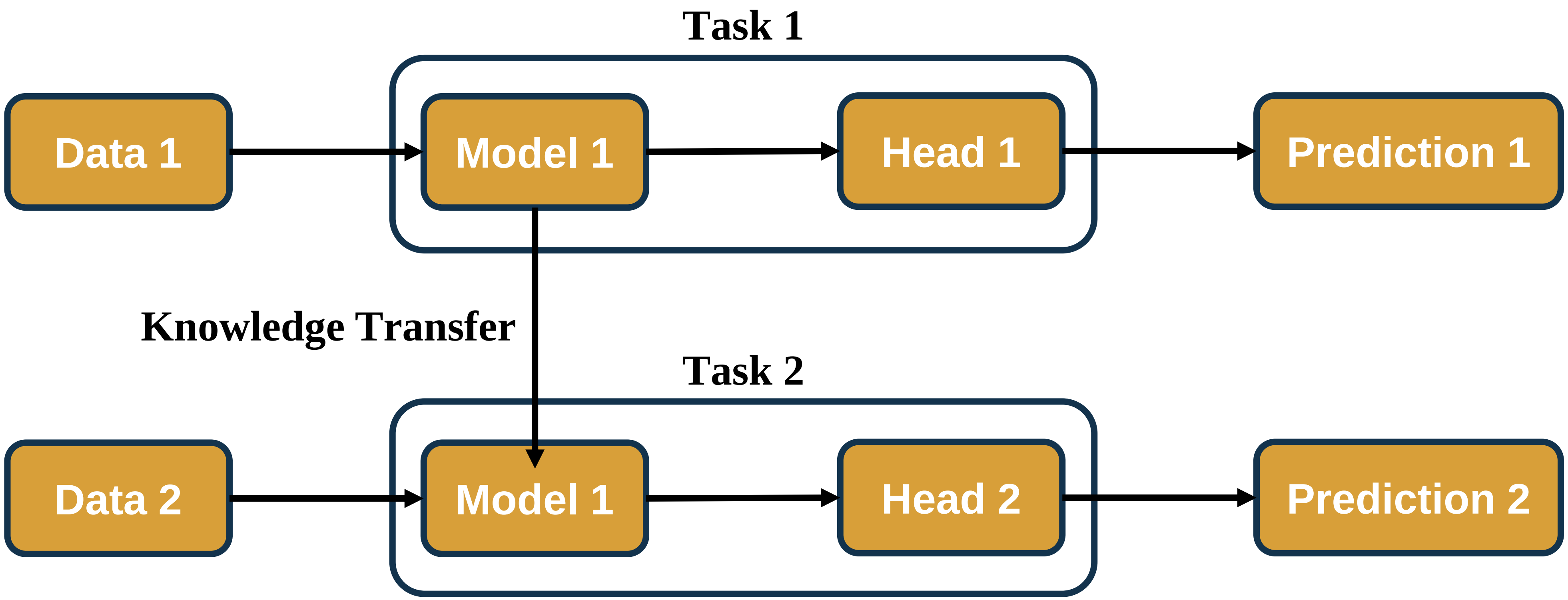
## Two-Stage Learning / MFCascadeCNN




# Parameters 662,513

# Special-purpose Learning Methods

## Transfer Learning



# Special-purpose Learning Methods

## Loss / MFE & MSFE

$$FPE = \frac{1}{N}\sum_{i=1}^{N}\sum_{n}\frac{1}{2}\left(d_n^{(i)} - y_n^{(i)}\right)^2$$

$$FNE = \frac{1}{P}\sum_{i=1}^{N}\sum_{n}\frac{1}{2}\left(d_n^{(i)} - y_n^{(i)}\right)^2$$

$$MFE = FPE + FNE$$

$$MSFE = FPE^2 + FNE^2$$

$MFE$: Mean False Error
$MSFE$: Mean Square False Error
$FPE$: Mean False Positive Error
$FNE$: Mean False Negative Error
$N$: the numbers of samples in negative class
$P$: the numbers of samples in positive class
$d_n^{(i)}$: the desired value of $i^{th}$ sample on $n^{th}$ neuron
$y_n^{(i)}$: the corresponding predicted value of $d_n^{(i)}$

# Special-purpose Learning Methods

## Optimizer / ImbSAM

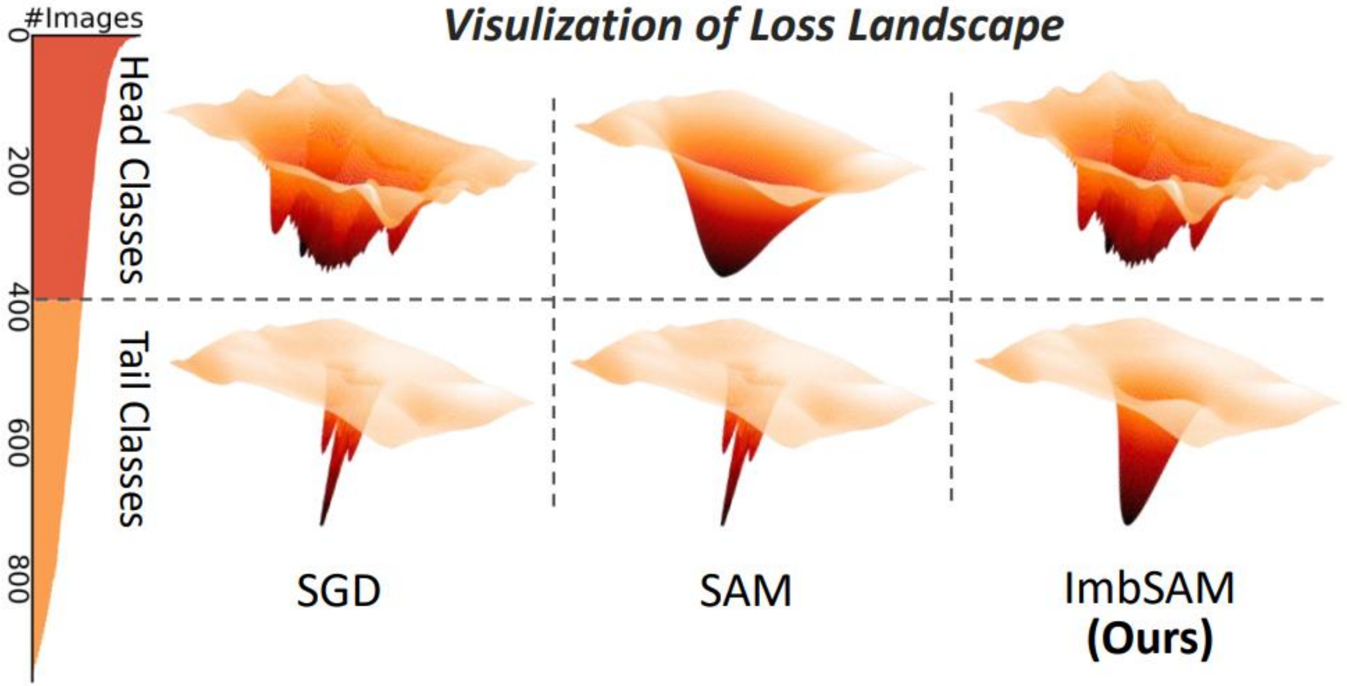



Figure 1: The visualization of separate loss landscape for *head* and *tail* classes in class-imbalanced recognition, optimized by SGD [6], SAM [17] and our ImbSAM respectively.

# Prediction Post-processing

## Introduction

Advantages:

- Not necessary to be aware of the user preference biases at learning time
- Be applied to different deployment scenarios without the need of re-learning the models
- Any standard learning tool can be used

Disadvantages:

- The models do not reflect the user preferences
- The models interpretability is meaningless

# Prediction Post-processing

## Thresholding

```
[2]
# Let's assume you have a classifier that outputs the following probabilities for a given datapoint x
# These are just example values for demonstration purposes
classifier_output_A = 0.3  # Probability that x belongs to Class A
classifier_output_B = 0.7  # Probability that x belongs to Class B

# Number of instances in each class
instances_A = 10
instances_B = 70

# Total number of instances
total_instances = instances_A + instances_B

# Calculating the prior probabilities for each class
prior_prob_A = instances_A / total_instances
prior_prob_B = instances_B / total_instances

# Adjusting the classifier's probabilities by the prior probabilities
adjusted_prob_A = classifier_output_A / prior_prob_A
adjusted_prob_B = classifier_output_B / prior_prob_B

# Normalizing the adjusted probabilities so that they sum up to 1
sum_adjusted_probs = adjusted_prob_A + adjusted_prob_B
normalized_prob_A = adjusted_prob_A / sum_adjusted_probs
normalized_prob_B = adjusted_prob_B / sum_adjusted_probs

(adjusted_prob_A, adjusted_prob_B), (normalized_prob_A, normalized_prob_B)
```

```
((2.4, 0.7999999999999999), (0.75, 0.25))
```

# References

## Notion

https://transparent-sesame-adf.notion.site/Imbalanced-4310cb6fbcfb4aa493145ef244c02f43

# Thanks!

**Any questions?**