Sampling Methods for Imbalance

Class imbalance is a common issue in machine learning where one class has significantly more samples than another. This can lead to biased models that favor the majority class. To address this, various **sampling methods** are used, primarily:

1. **Random Oversampling (ROS)**
2. **Random Undersampling (RUS)**
3. **Synthetic Minority Oversampling Technique (SMOTE)**
4. **Adaptive Synthetic Sampling (ADASYN)**
5. **NearMiss (Undersampling)**
6. **Cluster-Based Sampling**

Let's understand **Random Oversampling** and **Random Undersampling** with two detailed numerical examples.

Example 1: Random Oversampling (ROS)

Problem Statement:
Suppose we have a binary classification dataset with 1000 samples. The distribution is:

- **Class 0 (Majority class):** 900 samples
- **Class 1 (Minority class):** 100 samples

To balance the dataset, we use **Random Oversampling**, where we randomly duplicate minority class samples until both classes are equal.

Solution:

1. **Target Samples in Minority Class**
   To balance, we need **900 samples** in Class 1.
   - Currently, Class 1 has 100 samples.
   - We need **900 - 100 = 800** additional samples.
2. **Oversampling Process**
   - We randomly select samples from Class 1 and duplicate them.
   - Suppose we randomly select Sample IDs:

$$S1, S2, S3, ..., S100$$

     - We duplicate these multiple times until the count reaches 900.

- The new dataset will have:
  - **Class 0 (Majority):** 900 samples
  - **Class 1 (Minority after oversampling):** 900 samples

Impact:

- Balances the dataset without losing majority class information.
- Increases computational cost (since the dataset size grows).
- May lead to **overfitting** as the same minority samples are repeated.

Example 2: Random Undersampling (RUS)

Problem Statement:
Consider the same dataset:

- **Class 0 (Majority class):** 900 samples
- **Class 1 (Minority class):** 100 samples

Instead of oversampling, we use **Random Undersampling**, where we randomly remove samples from the majority class to balance the dataset.

Solution:

1. **Target Samples in Majority Class**
   To balance, we need **100 samples** in Class 0 (same as Class 1).
   - Currently, Class 0 has 900 samples.
   - We need to **remove 800 samples**.
2. **Undersampling Process**
   - We randomly select 800 samples from Class 0 and remove them.
   - Suppose we keep the following 100 samples:

$$S101, S302, S509, ..., S980$$

   - The new dataset will have:
     - **Class 0 (Majority after undersampling):** 100 samples
     - **Class 1 (Minority):** 100 samples

Impact:

- Reduces dataset size, making training faster.
- Prevents overfitting to minority class.
- **Loss of valuable information** due to removal of majority class samples.

Key Takeaways

Sampling Method	Approach	Advantages	Disadvantages
Random Oversampling (ROS)	Duplicates minority class samples	Balances class ratio, retains all data	Overfitting risk, increased dataset size
Random Undersampling (RUS)	Removes majority class samples	Faster training, prevents overfitting	Loss of important majority class information

Example 3: Synthetic Minority Oversampling Technique (SMOTE)

Problem Statement:
Consider a dataset with 1000 samples, where:

- **Class 0 (Majority class):** 900 samples
- **Class 1 (Minority class):** 100 samples

Instead of simply duplicating minority samples (**Random Oversampling**), SMOTE **generates synthetic samples** using interpolation techniques.

Solution:

1. **Determine the Target Samples in Minority Class**
   - To balance, we need **900 samples** in Class 1.
   - We currently have **100 samples**, so we need **800 synthetic samples**.
2. **SMOTE Process**
   - Select a random sample from Class 1.
   - Identify its **k-nearest neighbors** (commonly, **k = 5**).
   - Randomly select one of the neighbors and create a synthetic point between them using the formula:

     $$X_{\text{new}} = X_{\text{original}} + \lambda \times (X_{\text{neighbor}} - X_{\text{original}})$$

     where λ is a random number between 0 and 1.

Example Calculation:

- Assume we have two Class 1 samples:
  - $X_{\text{original}} = (2, 3)$
  - $X_{\text{neighbor}} = (4, 5)$
- Generate a synthetic sample using:

$$X_{\text{new}} = (2, 3) + 0.5 \times [(4, 5) - (2, 3)]$$

$$= (2, 3) + 0.5 \times (2, 2)$$

$$= (2, 3) + (1, 1) = (3, 4)$$

- This process is repeated until we generate **800 synthetic samples**.

Final Dataset Distribution:

- **Class 0:** 900 samples
- **Class 1 (after SMOTE):** 900 samples

Impact of SMOTE:
✅ Generates diverse synthetic samples, reducing overfitting.
✅ Preserves the **feature distribution** better than random oversampling.
❌ Can **create noise** if not tuned properly.
❌ **Does not address class overlap**, leading to potential misclassification.

Example 4: Adaptive Synthetic Sampling (ADASYN)

ADASYN is an extension of SMOTE but focuses on generating **more synthetic samples for hard-to-learn examples** rather than uniformly generating samples.

Problem Statement:
Consider the same dataset:

- **Class 0 (Majority class):** 900 samples
- **Class 1 (Minority class):** 100 samples

Solution:

1. **Identify Hard-to-Learn Samples**
   - Compute the **imbalance ratio**:

$$IR = \frac{\text{Majority Class Size}}{\text{Minority Class Size}} = \frac{900}{100} = 9$$

   - For each minority sample, calculate the **number of majority class neighbors (k-nearest neighbors)**.
   - More synthetic samples are generated for samples with a higher **majority-class presence** in their neighborhood.
2. **ADASYN Formula for Synthetic Data:**

$$X_{\text{new}} = X_{\text{original}} + \lambda \times (X_{\text{neighbor}} - X_{\text{original}})$$

   where λ is a weight **determined by class density** (i.e., more weight for hard-to-learn samples).

Example Calculation:

- Consider three Class 1 samples:

- $A = (2, 3)$ (has 4 majority neighbors)
- $B = (4, 5)$ (has 1 majority neighbor)
- $C = (6, 7)$ (has 0 majority neighbors)

- ADASYN assigns **higher synthetic sample generation** for A since it has more majority neighbors.
- **Fewer** samples are generated for C as it is already well-represented.

Final Dataset Distribution:

- **Class 0:** 900 samples
- **Class 1 (after ADASYN):** ~900 samples, but **denser near harder samples**.

Comparison: SMOTE vs ADASYN

Method	Approach	Strengths	Weaknesses
SMOTE	Interpolates synthetic samples	Prevents overfitting, retains diversity	May create unrealistic samples
ADASYN	Generates more samples in hard-to-learn regions	Focuses on misclassified areas	May reinforce noise

---

Final Thoughts

📌 **Use SMOTE** when you want **even distribution** and **more diverse samples**.
📌 **Use ADASYN** when the dataset has **challenging decision boundaries**.
📌 **For extremely imbalanced cases** (e.g., 99:1 ratio), **combine oversampling (SMOTE) with undersampling (NearMiss or RUS)**.

Example 5: NearMiss (Undersampling)

NearMiss is an **undersampling technique** that **selects only the most informative majority class samples**, rather than randomly removing them. This reduces data loss while balancing the dataset.

Problem Statement:
We have a dataset with:

- **Class 0 (Majority):** 900 samples
- **Class 1 (Minority):** 100 samples

Instead of randomly removing 800 samples from Class 0 (**Random Undersampling**), we apply **NearMiss**, which removes only the least informative samples.

Types of NearMiss

1. **NearMiss-1**: Keeps majority samples **closest to the minority class** (ensuring better decision boundary).
2. **NearMiss-2**: Keeps majority samples **far from other majority samples** (ensuring diversity).
3. **NearMiss-3**: Selects **majority samples with the nearest k-neighbors** from the minority class.

Solution (NearMiss-1 Example)

1. **Compute Distance Between Samples**
   - Consider three **majority class** samples:
     - $A = (2, 3)$, $B = (5, 6)$, $C = (10, 12)$
   - Consider a **minority class** sample:
     - $M = (3, 4)$
2. **Compute Euclidean Distance**
   - Distance between A and M:

$$d(A, M) = \sqrt{(3-2)^2 + (4-3)^2} = \sqrt{1+1} = \sqrt{2} \approx 1.41$$

   - Distance between B and M:

$$d(B, M) = \sqrt{(5-3)^2 + (6-4)^2} = \sqrt{4+4} = \sqrt{8} \approx 2.83$$

   - Distance between C and M:

$$d(C, M) = \sqrt{(10-3)^2 + (12-4)^2} = \sqrt{49+64} = \sqrt{113} \approx 10.63$$

3. **Undersampling Decision**
   - **Keep** samples closest to minority class → A, B
   - **Remove** distant samples → C

Final Dataset Distribution (After NearMiss-1)

- **Class 0 (Majority after undersampling):** 100 samples
- **Class 1 (Minority):** 100 samples

Impact of NearMiss

✅ Ensures the majority class **retains meaningful** samples.
✅ Preserves **decision boundaries** for better classification.
❌ Can **remove critical majority class samples**, leading to **underfitting**.

Example 6: Cluster-Based Sampling

Cluster-based sampling groups similar data points together, then selects representative samples from each group. This helps retain the dataset's overall structure while reducing class imbalance.

Problem Statement:
Consider a dataset with:

- **Class 0 (Majority):** 900 samples
- **Class 1 (Minority):** 100 samples

Instead of **randomly removing** majority samples, we apply **clustering (e.g., k-means)** to divide majority class samples into groups and retain representative samples from each.

Solution (k-Means Clustering)

1. **Apply k-Means Clustering on Class 0**
    - Suppose we set **k = 10** (dividing Class 0 into 10 clusters).
    - The **900 samples** are now grouped into **10 clusters**.
2. **Select Representative Samples**
    - From each cluster, we **randomly sample** 10 points.
    - This ensures that different **patterns and distributions** of Class 0 are preserved.

Final Dataset Distribution (After Clustering-Based Undersampling)

- **Class 0 (Majority after clustering-based undersampling):** 100 samples
- **Class 1 (Minority):** 100 samples

Impact of Cluster-Based Sampling

✅ Retains **diverse majority class samples**.
✅ Prevents **random information loss**.
❌ Computationally expensive compared to simple undersampling.

Final Comparison of Sampling Methods

Method	Approach	Strengths	Weaknesses
Random Oversampling (ROS)	Duplicates minority class samples	Prevents information loss	Overfitting risk
Random Undersampling (RUS)	Removes majority class samples	Faster training	Loss of valuable information
SMOTE	Interpolates between minority class samples	Reduces overfitting	May create synthetic noise
ADASYN	Focuses on hard-to-learn areas	Improves decision boundaries	May reinforce noise
NearMiss	Selects closest majority samples	Retains informative	May remove critical

Method	Approach	Strengths	Weaknesses
		data	samples
Cluster-Based Sampling	Groups majority samples, selects representatives	Preserves diversity	Computationally expensive

---

Final Thoughts

✅ **For small datasets** → Use **SMOTE** or **ADASYN**
✅ **For large imbalanced datasets** → Use **NearMiss** or **Cluster-Based Sampling**
✅ **For extreme imbalance (e.g., fraud detection, rare disease prediction)** → Use **a hybrid approach (SMOTE + NearMiss)**