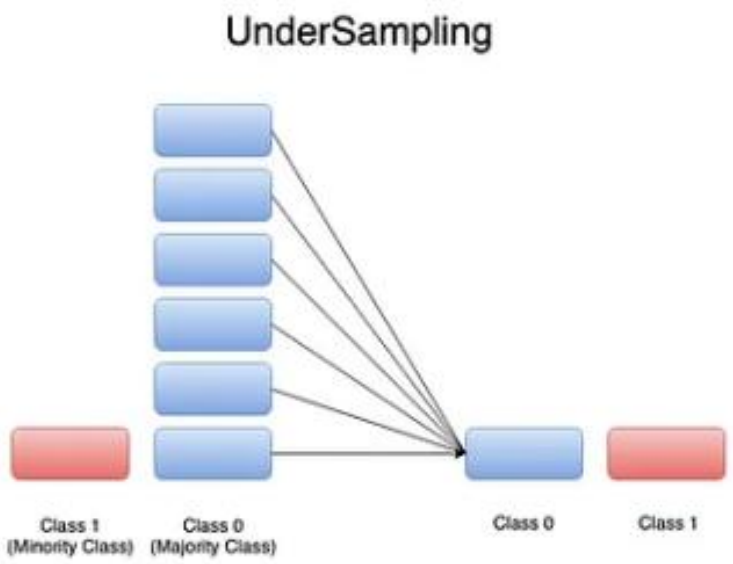
Explain Random Undersampling to handle Unbalanced Data



Random Undersampling is another straightforward technique to address class imbalance. It involves randomly removing instances from the majority class to reduce its representation in the training data, thereby balancing it with the minority class.

How it Works:

1. **Identify the Majority Class:** Determine the class(es) with a significantly higher number of instances.
2. **Determine the Target Ratio:** Decide on the desired ratio between the majority and minority classes after undersampling (e.g., 1:1, or a less extreme ratio).
3. **Randomly Select Instances for Removal:** Randomly pick instances from the majority class.
4. **Remove Selected Instances:** Delete the randomly selected majority class instances from the training dataset.
5. **Repeat:** Repeat steps 3 and 4 until the desired class distribution is achieved.

Example: Customer Churn Prediction

Let's say we are building a model to predict whether a customer will churn (Yes) or not (No). Our initial training dataset looks like this:

Customer ID	Contract Length (Months)	Monthly Charges	Total Charges	Churn (Target)
1	24	55.0	1320	No
2	1	70.0	70	Yes
3	72	80.0	5760	No
4	12	65.0	780	No
5	1	90.0	90	Yes
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
(4900 more "No" churn customers)	...	...	...	**No**

In this dataset, we have:

- **No Churn (Majority Class):** 4900 instances
- **Yes Churn (Minority Class):** 100 instances

This is a highly unbalanced dataset (49:1 ratio). A standard classifier trained on this data might be very good at predicting "No Churn" but poor at identifying customers who will actually churn.

Applying Random Undersampling:

To balance the data using random undersampling, let's say we want to achieve a 1:1 ratio between the "No Churn" and "Yes Churn" classes.

1. **Identify the Majority Class:** "No Churn" is the majority class.
2. **Determine the Target Ratio:** We want a 1:1 ratio, so we want approximately 100 instances of "No Churn."
3. **Randomly Select Instances for Removal:** We randomly select 4800 instances from the "No Churn" class.
4. **Remove Selected Instances:** We remove these 4800 randomly selected "No Churn" customers from our training dataset.

Resulting Dataset (Illustrative):

Customer ID	Contract Length (Months)	Monthly Charges	Total Charges	Churn (Target)
2	1	70.0	70	Yes
5	1	90.0	90	Yes
...	...	...	...	Yes
...	...	...	...	Yes
(100 "Yes" churn customers)	...	...	...	**Yes**
1	24	55.0	1320	No
4	12	65.0	780	No
...	...	...	...	No
...	...	...	...	No
(approximately 100 randomly selected "No" churn customers)	...	...	...	**No**

Now, the training dataset has a balanced class distribution (approximately 100 "Yes" and 100 "No" churn customers). This should help the classifier learn the patterns associated with churn more effectively.

Pros of Random Undersampling:

- **Simple to Implement:** It's a very basic and easy-to-code technique.
- **Reduces Training Time:** By decreasing the size of the majority class, it can significantly speed up the training process, especially for very large datasets.
- **Can Help with Memory Issues:** Reducing the number of majority class instances can alleviate memory problems when dealing with massive datasets.

Cons of Random Undersampling:

- **Potential for Information Loss:** Randomly removing majority class instances can discard potentially important information and patterns that the classifier could have learned. This can lead to a model with lower overall performance compared to using the full dataset (if the imbalance wasn't too severe).
- **Can Introduce Bias:** If the randomly selected subset of the majority class is not truly representative of the entire majority class, it can introduce bias into the model.

When to Consider Random Undersampling:

- When the dataset is very large, and reducing the number of majority class instances is necessary for computational reasons.
- When you have a massive amount of data for the majority class, and losing some instances might not significantly impact the learning process.
- As a quick and dirty way to try and address imbalance, often used as a baseline before trying more sophisticated techniques.

Important Note: Random undersampling should be used with caution, as it can lead to a loss of valuable information. It's often beneficial to explore other techniques like oversampling or a combination of both, or algorithm-level methods, to see if they yield better results without discarding potentially useful data. Techniques like informed undersampling (e.g., NearMiss, Tomek Links) try to address the information loss issue by strategically selecting which majority class instances to remove.