Dealing with Imbalanced Data

Target Class Imbalance

- ML models tend to favor the majority class.
- Consequences:
  - Reduced model performance, especially for the minority class.
  - Increased likelihood of overfitting to the majority class.

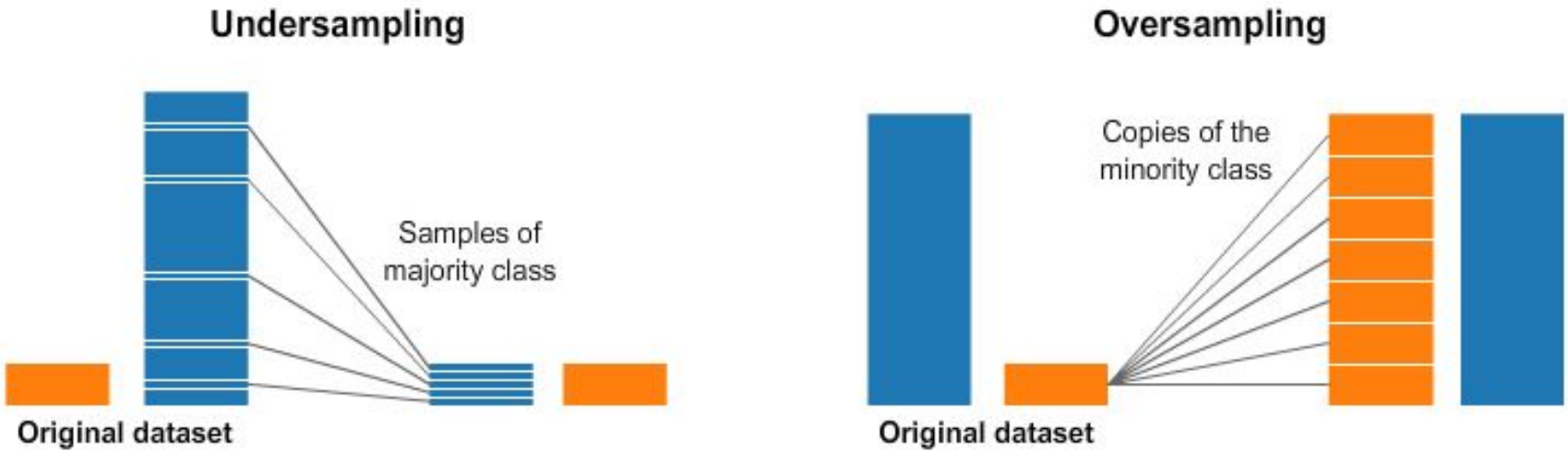
Target Class Imbalance

- ML models tend to favor the majority class.
- Consequences:
  - Reduced model performance, especially for the minority class.
  - Increased likelihood of overfitting to the majority class.
- Common techniques:
  - Give more weight to the minority class.
  - Equilibrate the number of samples of each class.

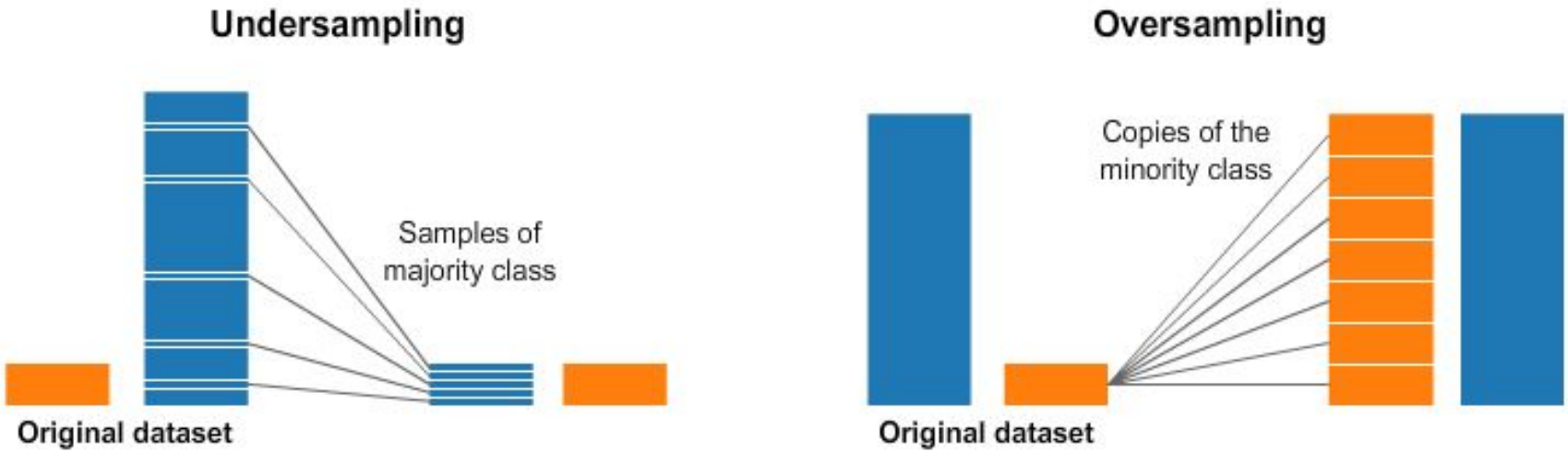
Class Weight

- In sklearn, most classification algorithms have a `class_weight` hyperparameter.
- It modifies the loss function to change the weights of the classes.
- Two options:
  - `"balanced"` automatically adjusts weights inversely proportional to class frequencies.
    - The rarer the class, the more weight it gets.
  - Dict of the form: `{0: 1, 1: 3}`
    - In this example, class `1` is considered 3 times as important as class `0`.

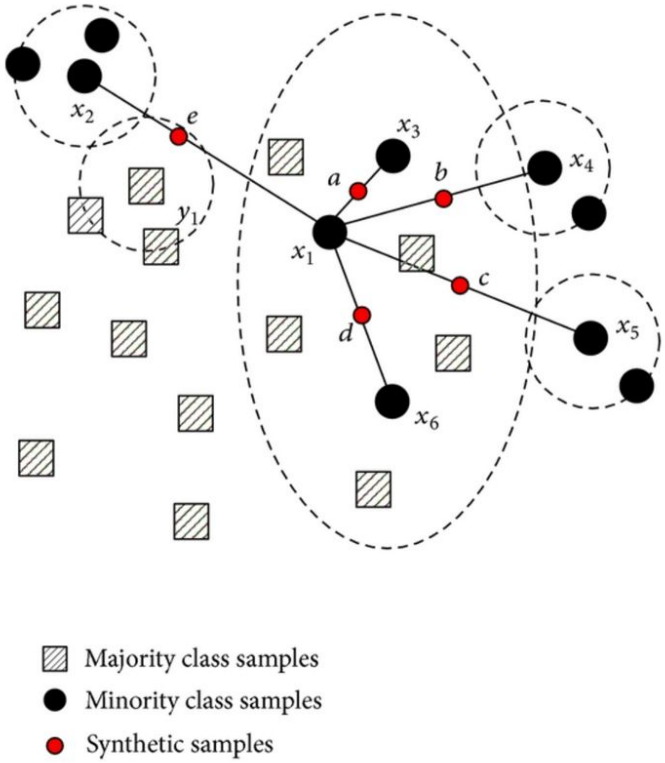
Undersampling and Oversampling



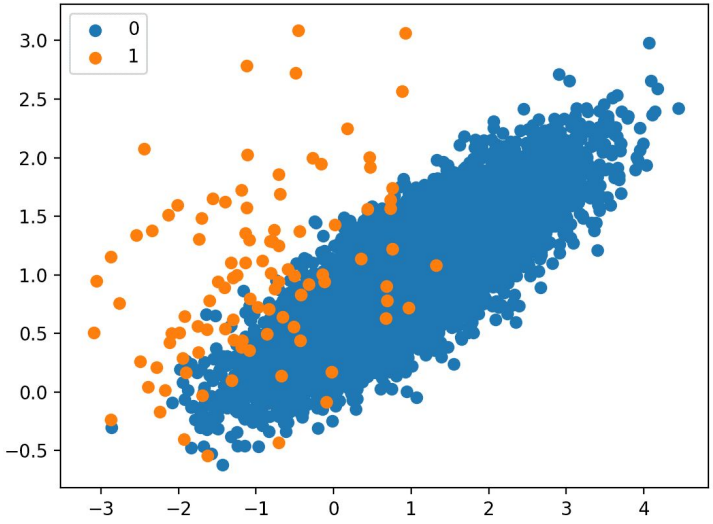
Random Undersample or Oversample



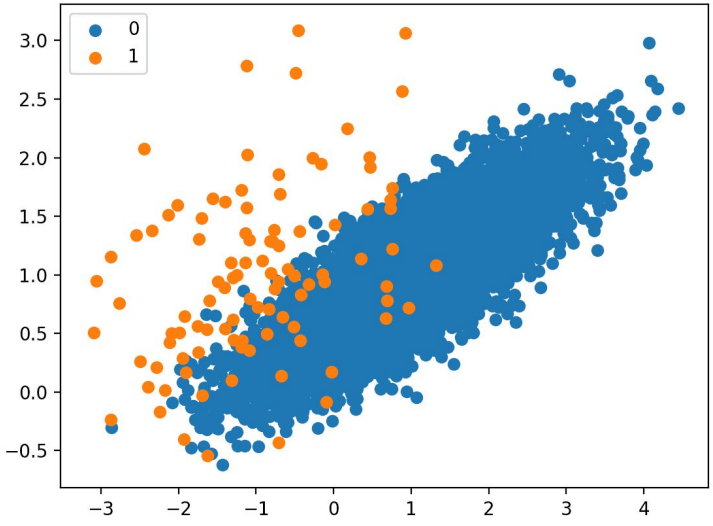
Oversampling: SMOTE

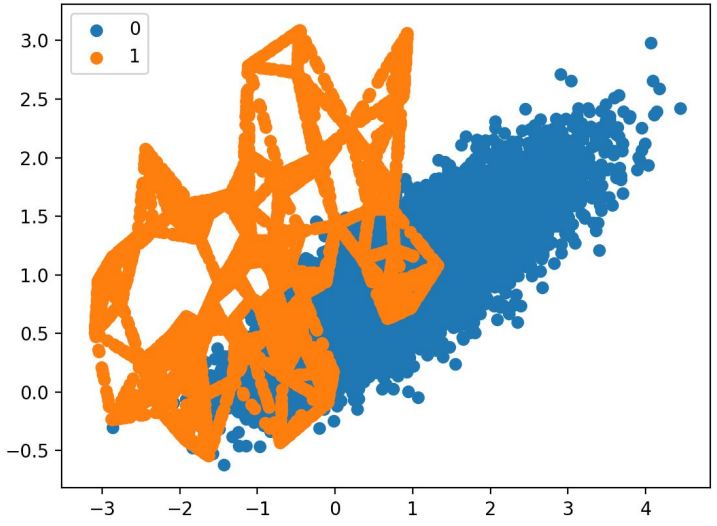


SMOTE



SMOTE


0
1
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−3
−2
−1
0
1
2
3
4


0
1
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−3
−2
−1
0
1
2
3
4

Imblearn

- `imblearn` package has functionalities to balance classes.
- By default, sklearn's `Pipeline` doesn't allow to modify the target `y`.
- So we must use `imblearn Pipeline` for balancing.

Imblearn: Random Undersampling Example

```
from sklearn.linear_model import LogisticRegression
from imblearn.pipeline import Pipeline as ImbPipeline
from imblearn.under_sampling import RandomUnderSampler


lr_pipe = ImbPipeline([
    ('RUS', RandomUnderSampler()),
    ('LR', LogisticRegression())
])
```

Imblearn: SMOTE Example

```
from sklearn.linear_model import LogisticRegression
from imblearn.pipeline import Pipeline as ImbPipeline
from imblearn.over_sampling import SMOTE


lr_pipe = ImbPipeline([
    ('SMOTE', SMOTE(random_state=42)),
    ('LR', LogisticRegression())
])
```

Undersampling and Oversampling combination

- Examples:
  - Undersampling and Oversampling:
    - Undersampling a percentage of the majority class.
    - Then Oversampling the remaining percentage of the minority class.

Balanced Random Forest

- Random undersampling each tree.
- There is a `BalancedRandomForest` implementation in `imblearn.`