Using Random Under-Sampling When observations from the majority class are eliminated until the majority and minority classes are balanced, this is known as undersampling.

In [10]:

```
from sklearn.utils import resample
from sklearn.datasets import make_classification
import pandas as pd

#making DataFrame having 100 dummy samples with 4 features
#Divided in 2 classes in a ratio of 80:20
X, y = make_classification(n_classes=2, weights=[0.8,0.2],
                           n_features = 4, n_samples = 100,
                           random_state = 42)

df = pd.DataFrame(X, columns=['feature_1',
                              'feature_2',
                              'feature_3',
                              'feature_4'])
df['balance']=y
print(df)

#let df represent the dataset dividing majority and minority classes
df_major = df[df.balance == 0]
df_minor = df[df.balance == 1]


#upsampling mino class
df_minor_sample = resample(df_minor,
                           #upsapmle wtih replacement
                           replace=True,

                           #num to match maj class
                           n_samples=80,
                           random_state=42)

#combine maj and unxamp mino class
df_sample = pd.concat([df_major, df_minor_sample])

#disp count of data ponits in both class
print(df_sample.balance.value_counts())
```

```
    feature_1  feature_2  feature_3  feature_4  balance
0   -1.053839  -1.027544  -0.329294   0.826007        1
1    1.569317   1.306542  -0.239385  -0.331376        0
2   -0.658926  -0.357633   0.723682  -0.628277        0
3   -0.136856   0.460938   1.896911  -2.281386        0
4   -0.048629   0.502301   1.778730  -2.171053        0
..        ...        ...        ...        ...      ...
95  -2.241820  -1.248690   2.357902  -2.009185        0
96   0.573042   0.362054  -0.462814   0.341294        1
97  -0.375121  -0.149518   0.588465  -0.575002        0
98   1.042518   1.058239   0.461945  -0.984846        0
99  -0.121203  -0.043997   0.204211  -0.203119        0

[100 rows x 5 columns]
0    80
1    80
Name: balance, dtype: int64
```

Using RandomOverSampler:

In [13]:

```
from imblearn.over_sampling import RandomOverSampler
from sklearn.datasets import make_classification

#making dataset having 100 dummy samples with 4 features
#divided in 2 classes in a ratio of 80:20

X, y = make_classification(n_classes=2, weights=[0.8,0.2],
                           n_features = 4, n_samples = 100,
                           random_state = 42)
#printinf numb of samples in each calss before oversampling

t = [(d) for d in y if d == 0]
s= [(d) for d in y if d == 1]

print('Before over sampling')
print('samples in class 0:', len(t))
print('samples in class 1:', len(s))

#oversampling mino class

OverS = RandomOverSampler(random_state=42)
#fit predictor(x) and targ (y) using fit_resample()

X_over, Y_over = OverS.fit_resample(X, y)

#printing numb of samp in each ca;ss after over sampling

t = [(d) for d in Y_over if d == 0]
s= [(d) for d in Y_over if d == 1]

print('After over sampling')
print('samples in class 0:', len(t))
print('samples in class 1:', len(s))

```

```
Before over sampling
samples in class 0: 80
samples in class 1: 20
After over sampling
samples in class 0: 80
samples in class 1: 80
```

```
Random under sampling with imblearn
```

In [17]:

```
from imblearn.under_sampling import RandomUnderSampler
from sklearn.datasets import make_classification
#making dataset having 100 dummy samples with 4 features
#divided in 2 classes in a ratio of 80:20

X, y = make_classification(n_classes=2, weights=[0.8,0.2],
                           n_features = 4, n_samples = 100,
                           random_state = 42)
#printinf numb of samples in each calss before undersampling

t = [(d) for d in y if d == 0]
s= [(d) for d in y if d == 1]

print('Before under sampling')
print('samples in class 0:', len(t))
print('samples in class 1:', len(s))


#downsampling mino class

UnderS = RandomUnderSampler(random_state=42, replacement=True)
#fit predictor(x) and targ (y) using fit_resample()

X_under, Y_under = UnderS.fit_resample(X, y)

#printing numb of samp in each ca;ss after over sampling

t = [(d) for d in Y_under if d == 0]
s= [(d) for d in Y_under if d == 1]

print('After under sampling')
print('samples in class 0:', len(t))
print('samples in class 1:', len(s))
```

```
Before under sampling
samples in class 0: 80
samples in class 1: 20
After under sampling
samples in class 0: 20
samples in class 1: 20
```

In [ ]:

```
1
```