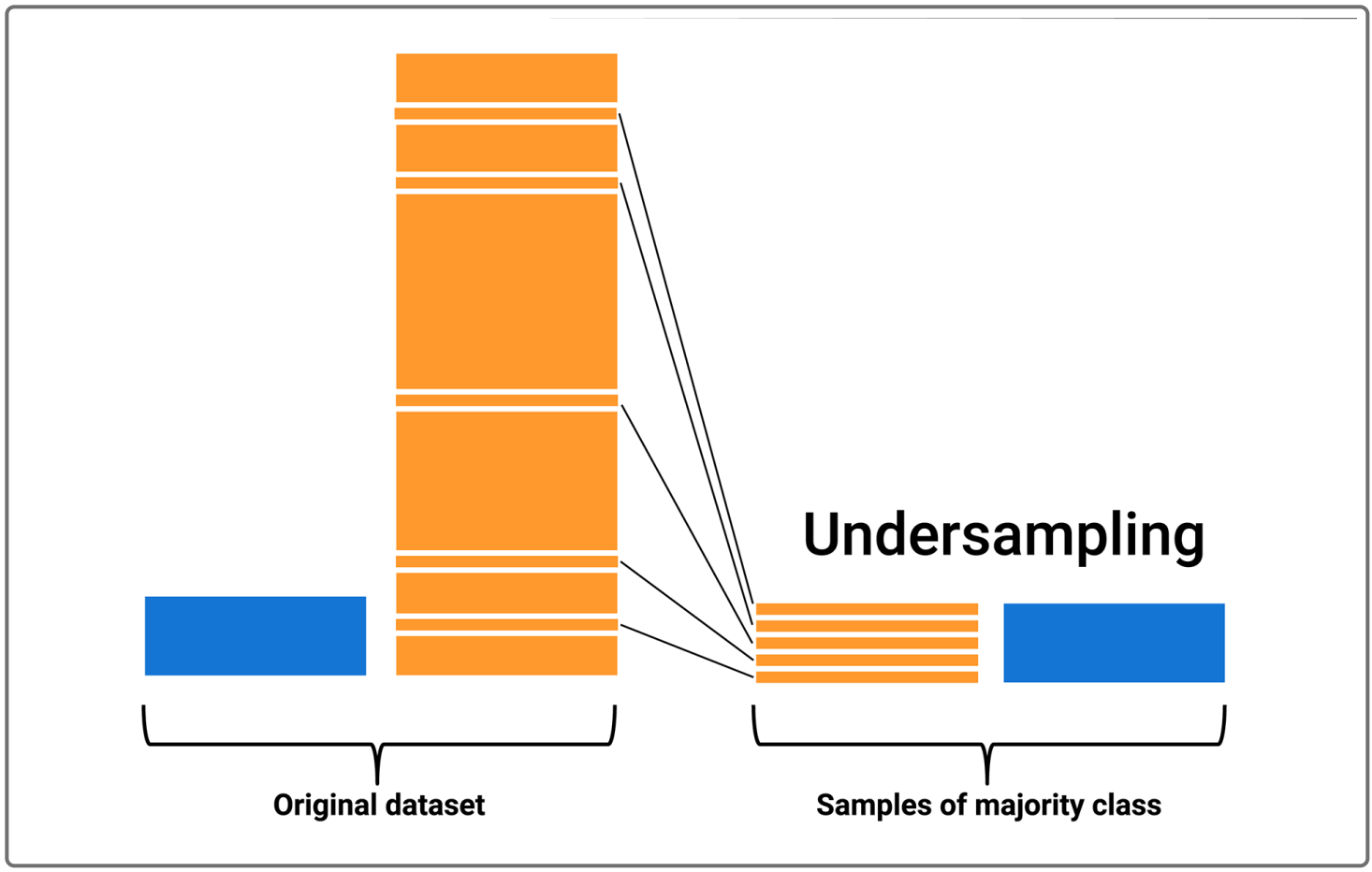
17.10.2

Undersampling

You've learned that in oversampling, the smaller class is resampled to make it larger. Undersampling, in contrast, takes the opposite tack. Jill recommends that you look at random undersampling as well as a synthetic approach.

Undersampling is another technique to address class imbalance. Undersampling takes the opposite approach of oversampling. Instead of increasing the number of the minority class, the size of the majority class is decreased.



Keep in mind that both oversampling and undersampling involve tradeoffs. Oversampling addresses class imbalance by duplicating or mimicking existing data. In contrast, undersampling only uses actual data. On the other hand, undersampling involves loss of data from the majority class. Furthermore, undersampling is practical only when there is enough data in the training set. There must be enough usable data in the undersampled majority class for a model to be useful.

We'll discuss two approaches to undersampling: random and cluster centroid. Both are similar to the oversampling methods we've seen. Download the files and let's look at the examples together.

[Download 17-10-2-undersampling.zip (https://2u-data-curriculum-team.s3.amazonaws.com/dataviz-online/module_17/17-10-2-undersampling.zip)]

Random Undersampling

In random undersampling, randomly selected instances from the majority class are removed until the size of the majority class is reduced, typically to that of the minority class. The dataset used in this example contains information on credit card default.

```
import pandas as pd
from path import Path
from collections import Counter

data = Path('../Resources/cc_default.csv')
df = pd.read_csv(data)
df.head()
```

	ID	ln_balance_limit	sex	education	marriage	age	default_next_month
0	1	9.903488	1	2	0	24	1
1	2	11.695247	1	2	1	26	1
2	3	11.407565	1	2	1	34	0
3	4	10.819778	1	2	0	37	0
4	5	10.819778	0	2	0	57	0

The following legend explains the values used in the columns:

- ln_balance_limit: maximum balance limit on a card
- sex: 1 = female, 0 = sex
- education: 1 = graduate school, 2 = university, 3 = high school, 4 = others
- marriage: 1 = married, 0 = single
- age: age of credit card holder
- default_next_month: 1 = yes, 0 = no

The target variable, `default_next_month`, describes whether a credit card holder defaults during the next month. The features are all columns in the dataset, except `ID` and `default_next_month`, with the former not a useful predictor of default status and the latter the target. We can drop these columns as we have done before, or we can use a list comprehension, as shown below.

```
x_cols = [i for i in df.columns if i not in ('ID', 'default_next_month')]
X = df[x_cols]
y = df['default_next_month']
```

The rest of the code used for undersampling is very similar to that used for oversampling. We first split the dataset into training and testing sets.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)
```

Next, we use `imblearn`'s `RandomUnderSampler` module to train the `RandomUnderSampler` instance, then undersample the majority class. Counting the class verifies that both classes are the same size.

```
from imblearn.under_sampling import RandomUnderSampler
ros = RandomUnderSampler(random_state=1)
X_resampled, y_resampled = ros.fit_resample(X_train, y_train)
Counter(y_resampled)
```

A `LogisticRegression` model will be used again on the dataset. We first train it.

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(solver='lbfgs', random_state=1)
model.fit(X_resampled, y_resampled)
```

Then, we make predictions and generate a `confusion_matrix`.

```
from sklearn.metrics import confusion_matrix
y_pred = model.predict(X_test)
confusion_matrix(y_test, y_pred)

array([[3732, 2100],
[ 740, 928]])
```

We calculate the `balanced_accuracy_score`, which is 0.598.

```
from sklearn.metrics import balanced_accuracy_score
balanced_accuracy_score(y_test, y_pred)
```

Finally, we print the classification report.

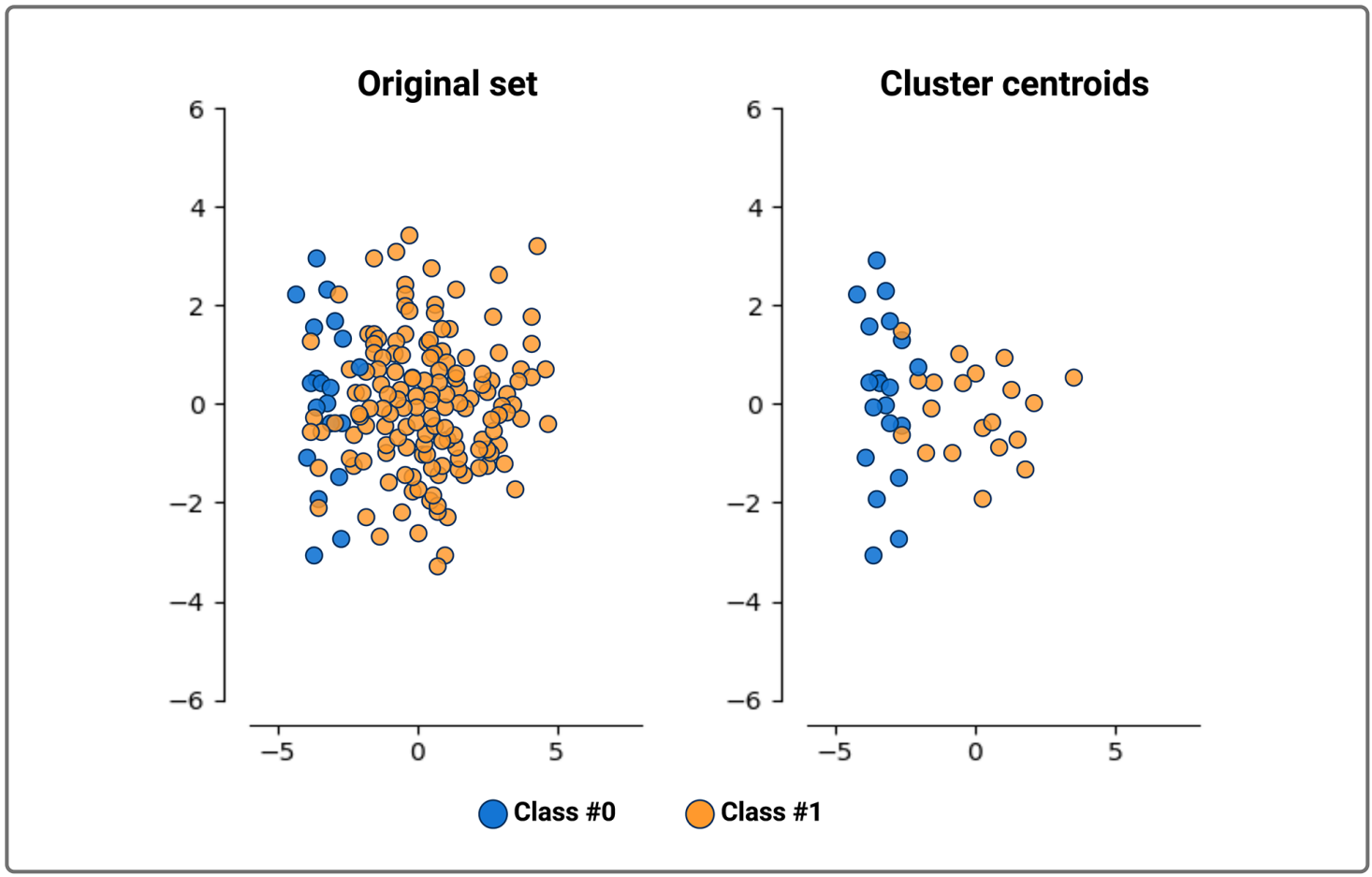
```
from imblearn.metrics import classification_report_imbalanced
print(classification_report_imbalanced(y_test, y_pred))
```

	pre	rec	spe	f1	geo	iba	sup
0	0.83	0.72	0.47	0.77	0.58	0.35	5832
1	0.32	0.47	0.72	0.38	0.58	0.33	1668
avg / total	0.71	0.66	0.53	0.68	0.58	0.34	7500

Generally, the results are unimpressive, especially for predicting defaults. Let's see whether another undersampling technique will improve the metrics.

Cluster Centroid Undersampling

Cluster centroid undersampling is akin to SMOTE. The algorithm identifies clusters of the majority class, then generates synthetic data points, called centroids, that are representative of the clusters. The majority class is then undersampled down to the size of the minority class.



To implement this technique in Python, we'll use `imblearn`'s `ClusterCentroids` module. The process is much the same as before:

1. Fit and resample the training data.
2. Train a logistic regression model.
3. Create predictions.
4. Assess the results.

NOTE

Some of these steps are computationally intensive and may take several minutes to complete.

First, instantiate the resampling module and use it to resample the data.

```
from imblearn.under_sampling import ClusterCentroids
cc = ClusterCentroids(random_state=1)
X_resampled, y_resampled = cc.fit_resample(X_train, y_train)
```

Then instantiate and train a logistic regression model.

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(solver='lbfgs', random_state=1)
model.fit(X_resampled, y_resampled)
```

Next, generate the metrics.

```
from sklearn.metrics import confusion_matrix
y_pred = model.predict(X_test)
confusion_matrix(y_test, y_pred)

from sklearn.metrics import balanced_accuracy_score
balanced_accuracy_score(y_test, y_pred)

from imblearn.metrics import classification_report_imbalanced
print(classification_report_imbalanced(y_test, y_pred))
```

	pre	rec	spe	f1	geo	iba	sup
0	0.82	0.48	0.64	0.60	0.55	0.30	5832
1	0.26	0.64	0.48	0.37	0.55	0.31	1668
avg / total	0.70	0.51	0.61	0.55	0.55	0.30	7500

These results are worse than those from random undersampling! This underscores an important point: While resampling can attempt to address imbalance, it does not guarantee better results.

SKILL DRILL

Perform oversampling on this dataset and compare results. Which resampling technique yields better results?