GJU
الجامعة الألمانية الأردنية
German Jordanian University

SEEIT

Balanced and Imbalanced Datasets

Rana Husni

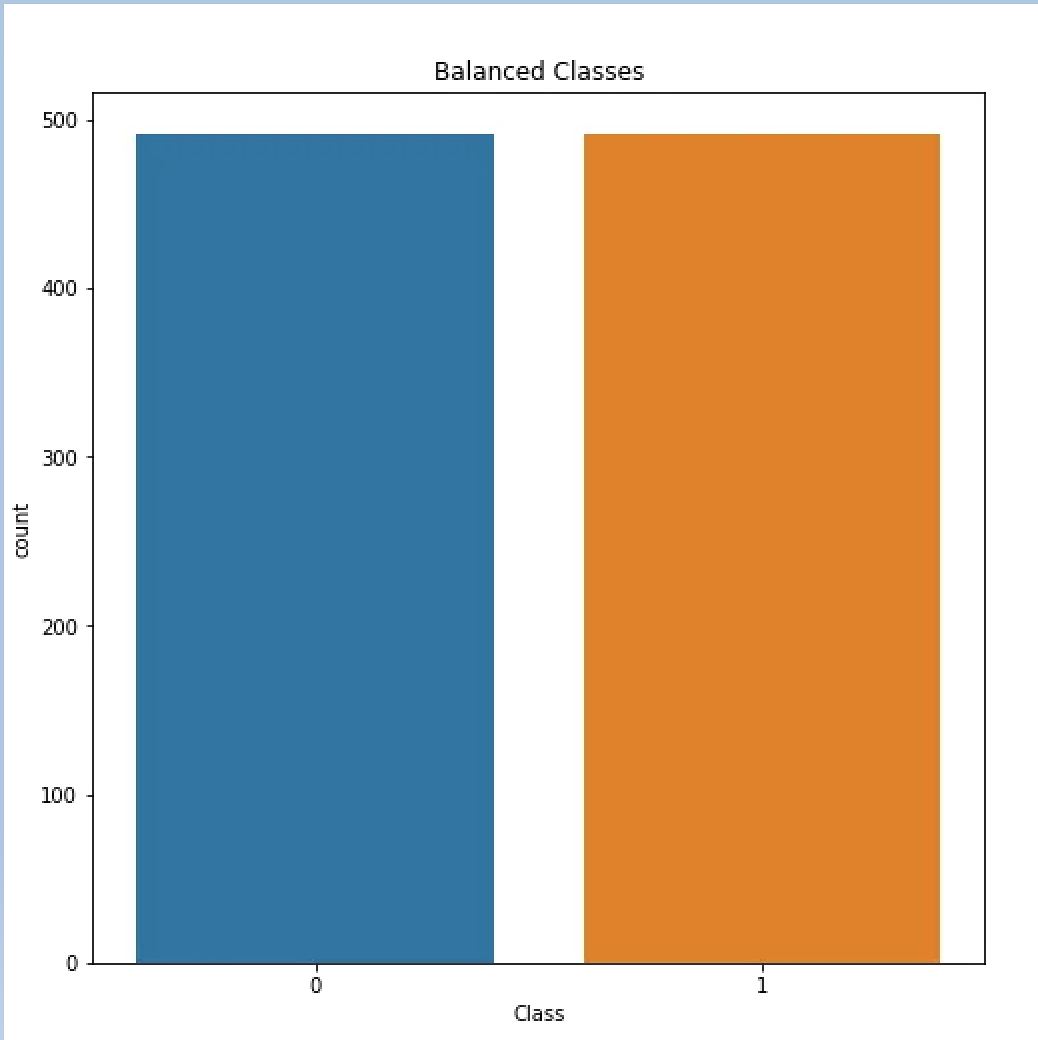
الجامعة الألمانية الأردنية
German Jordanian University

Balanced Dataset

SEEIT

Before giving you the definition of Balanced dataset let me give you an example for your better understanding, lets assume I have a dataset with thousand data points and I name it “N”. So now N = 1000 data points, & N have two different classes one is N1 and another one is N2. Inside the N1 there have 580 data points and inside the N2 there have 420 data points. N1 have positive (+Ve) data points and N2 have negative (-Ve) data points. So we can say that the number of data points of N1 and N2 is almost similar than each other. So then I can write N1 ~ N2. Then it is proved that N is a Balanced Dataset.

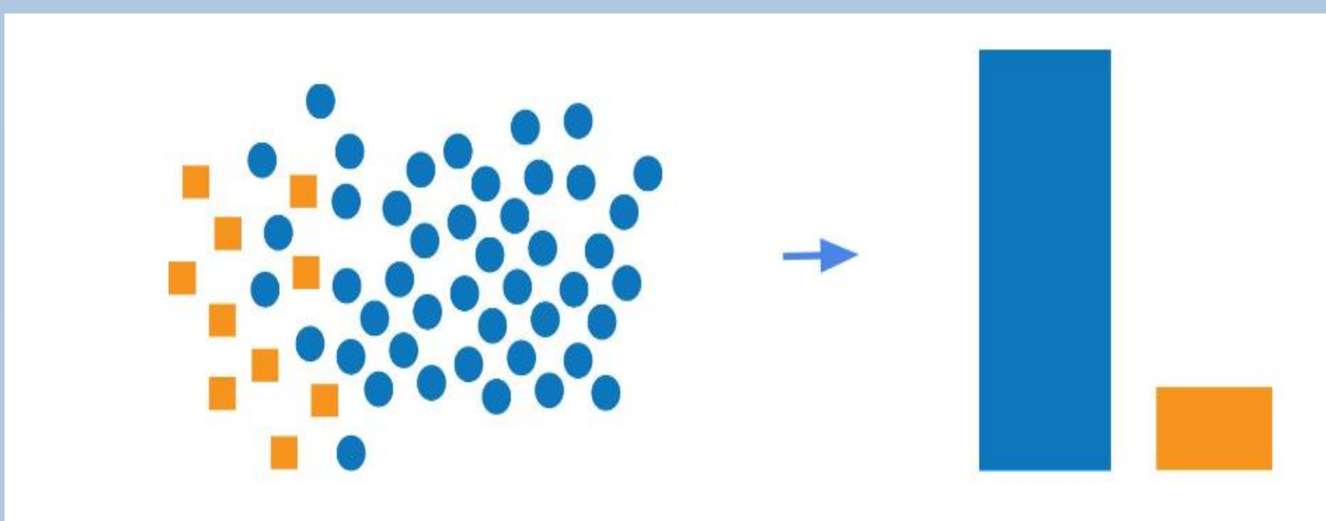
A balanced dataset is the one that contains an equal or almost equal number of samples from the positive and negative classes.



Rana Husni

Imbalanced Dataset

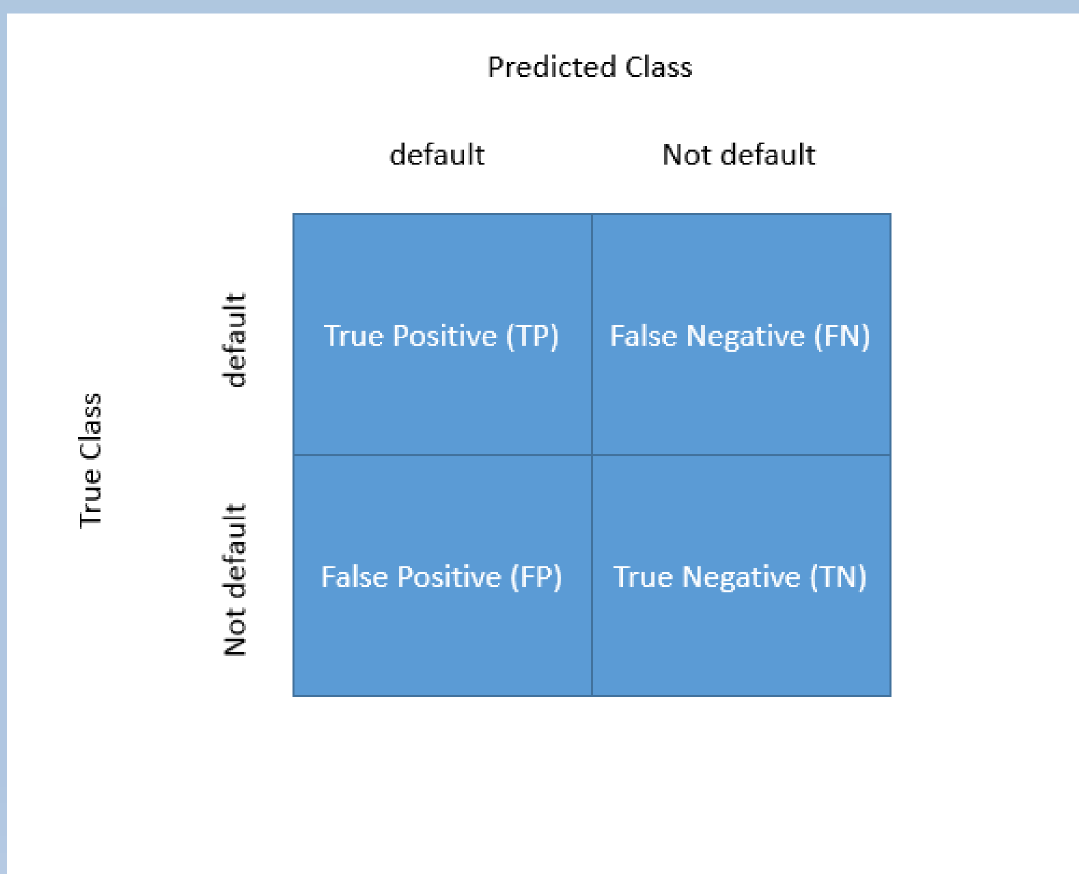
Before giving you the definition of Imbalanced dataset let me give you an example for your better understanding, lets assume I have a dataset with thousand data points and I name it “N”. So now N = 1000 data points, & N have two different classes one is N1 and another one is N2. Inside the N1 there have 900 data points and inside the N2 there have 100 data points. N1 have positive (+Ve) data points and N2 have negative (-Ve) data points. So we can say that the number of data points of N1 and N2 is not similar than each other. So then I can write N1 ≠ N2, then it is proved that N is an Imbalanced Dataset.



Why Accuracy is Bad Metric for Imbalanced Datasets

Consider a loan default prediction problem which has a total of 1000 data points out of which 100 are ‘default’ and remaining 900 are ‘Not default’. The ratio of ‘default’ to ‘Not default’ is 1:9. This is an imbalanced dataset.

Confusion Matrix for two class classification problem



Let us consider a dumb model that predicts ‘Not default’ for all data points. Below shows the confusion matrix

		Predicted Class	
		default	Not default
True Class	default	0	100
	Not default	0	900

Below shows the accuracy

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

So, the accuracy of our dumb model is 90%. On the face of it, 90% accuracy seems very good (which is still subjective) but no one deploys this model in production.

Let us calculate the F1-score for the above model.

$$F1 - score = \frac{2 * Precision * Recall}{Precision + Recall}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

Precision tells us that, of all the predicted values, how many of them are actually correct. Recall tells us that, of all the true values, how many of them are correctly predicted. F1-score is the harmonic mean of Precision and Recall. One important thing about harmonic mean is that it will be closer to the smaller value.

Note: In our case, ***correct*** implies ***default*** class

As TP = 0, F1-score, Precision and Recall will be 0. So, our 90% accurate dumb model has F1-score of zero.

Now, let us flip a coin and predict default when HEADS and Not default when TAILS. So, ideally we should get the below confusion matrix:

		Predicted Class	
		default	Not default
True Class	default	50	50
	Not default	450	450

Now, the accuracy is 50%, while Precision is 0.1, Recall is 0.5 and F1-score is 0.1667. We can see that F1-score is closer to smaller value, i.e., Precision.

Let us consider some model (namely ***ML model***) with the below confusion matrix:

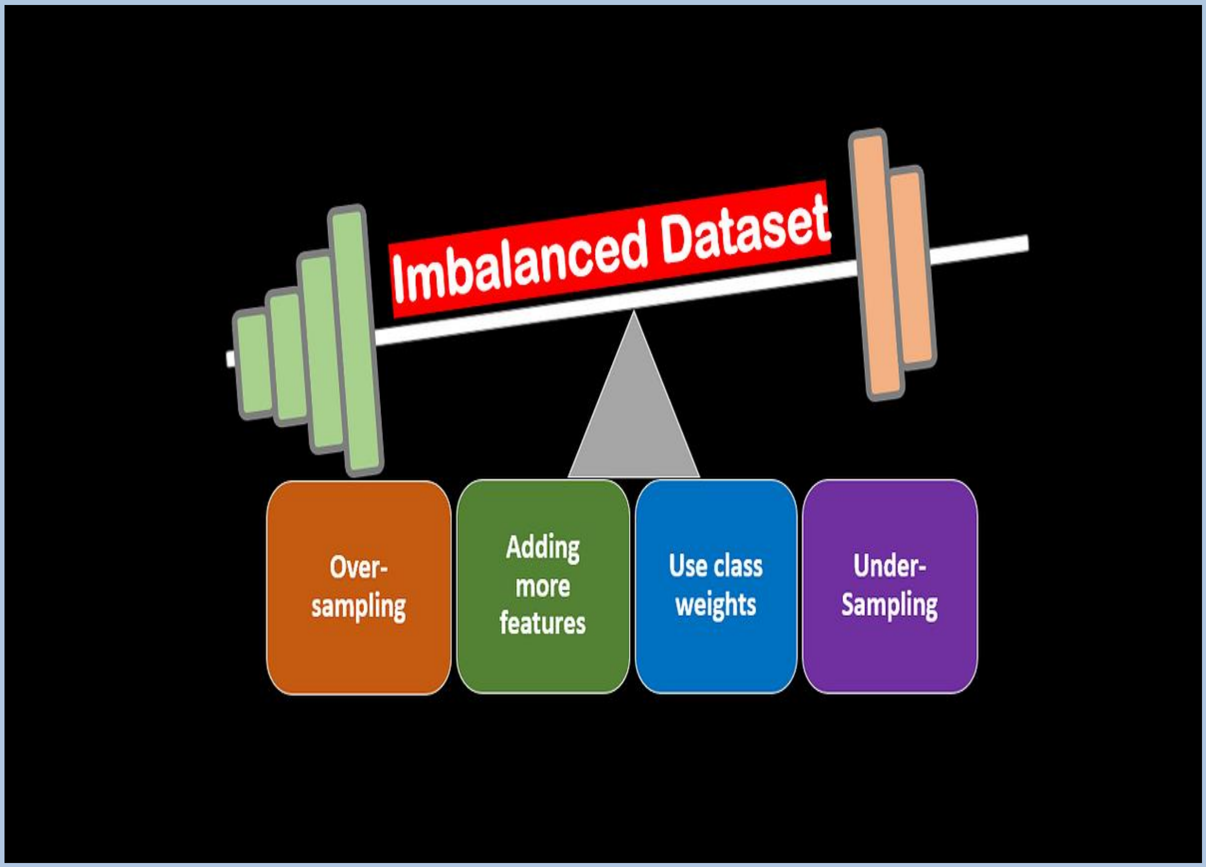
		Predicted Class	
		default	Not default
True Class	default	85	15
	Not default	50	850

For the ***ML model***, Precision is 0.629, Recall is 0.85 and F1-score is 0.723.

How to handle Imbalanced Dataset:

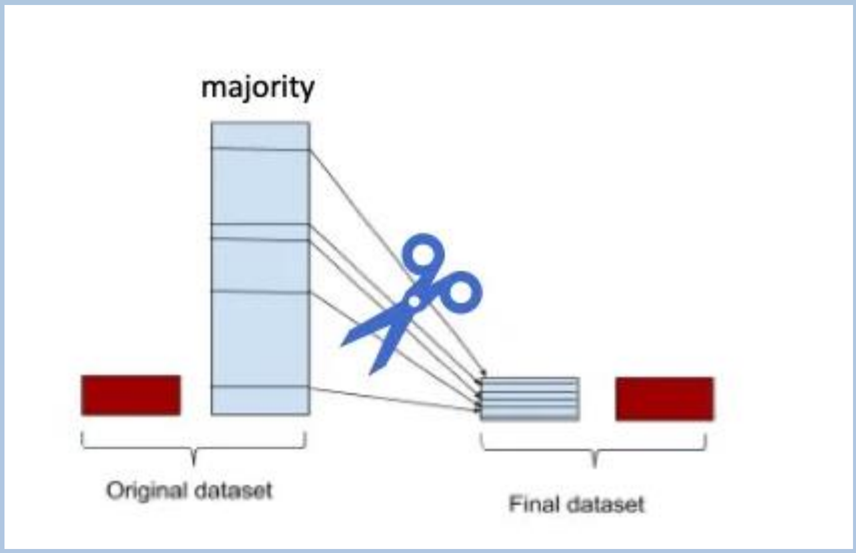
Well there have some few methods to handle an Imbalanced Dataset but there also have some problems, I will briefly explain all of them in below, there have two different methods to handle an Imbalanced dataset.

- Under-Sampling
- Over-Sampling



Under-Sampling :

Let assume we have a dataset "N" with 1000 data points. And 'N' have two class one is n1 and another one is n2. These two classes have two different reviews Positive and Negative. Here n1 is a positive class (+Ve) and have 900 data points and n2 is a negative class (-Ve) and have 100 data points, so we can say n1 is a majority class because n1 have big amount of data points and n2 is a minority class because n2 have less number of data points. For handle this Imbalanced dataset we will create a new dataset called N'. Here we will take all (100)n2 datapoints as it is and we will take randomly (100)n1 datapoints and put into the dataset called N'. This is a sampling trick and its called Under-Sampling.



- **Disadvantages of Under-Sampling:**

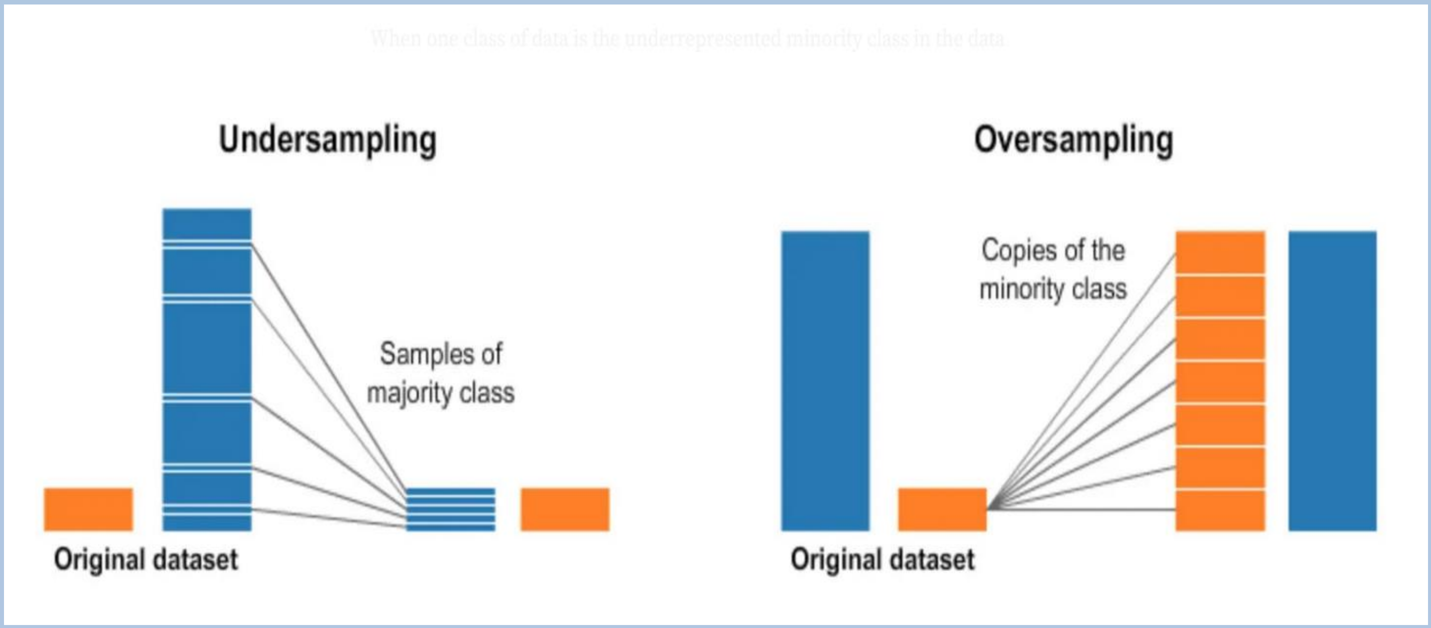
Before Under-Sampling we had 1000 data points in N and after Under-Sampling we had only 200 data points in N'. Now we have some data points, and we have thrown around 80% of data points which is not good for getting a good model because 80% of the datasets is also an 80% important information.

GJU
الجامعة الألمانية الأردنية
German Jordanian University

SEEIT

Over-Sampling:

When one class of data is the underrepresented minority class in the data sample, oversampling techniques may be used to duplicate these results for a more balanced amount of positive results in training. Oversampling is used when the amount of data collected is insufficient. A popular oversampling technique is SMOTE (Synthetic Minority Over-sampling Technique), which creates synthetic samples by randomly sampling the characteristics from occurrences in the minority class.



Rana Husni

Breast Cancer dataset

```
import numpy as np
from sklearn.datasets import make_classification
from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler
from collections import Counter
from sklearn.datasets import load_breast_cancer
data = load_breast_cancer()
X=data.data
y= data.target
print("Original class distribution:", Counter(y))

# Oversampling using RandomOverSampler
oversample = RandomOverSampler(sampling_strategy='minority')
X_over, y_over = oversample.fit_resample(X, y)
print("Oversampled class distribution:", Counter(y_over))

 # Undersampling using RandomUnderSampler
undersample = RandomUnderSampler(sampling_strategy='majority')
X_under, y_under = undersample.fit_resample(X, y)
print("Undersampled class distribution:", Counter(y_under))
```

```
Original class distribution: Counter({1: 357, 0: 212})
Oversampled class distribution: Counter({0: 357, 1: 357})
Undersampled class distribution: Counter({0: 212, 1: 212})
```