<https://www.analyticsvidhya.com/blog/2020/10/improve-class-imbalance-class-weights/>

When the class\_weights = ‘balanced’, the model automatically assigns the class weights inversely proportional to their respective frequencies.

To be more precise**,**the formula to calculate this is:

wj=n\_samples / (n\_classes \* n\_samplesj)

Here,

* wj is the weight for each class(j signifies the class)
* n\_samplesis the total number of samples or rows in the dataset
* n\_classesis the total number of unique classes in the target
* n\_samplesjis the total number of rows of the respective class

For our heart stroke example:

n\_samples=  43400,  n\_classes= 2(0&1), n\_sample0= 42617, n\_samples1= 783

Weights for class 0:

w0=  43400/(2\*42617) = 0.509

Weights for class 1:

w1= 43400/(2\*783) = 27.713

I hope this makes things more clear that how class\_weight = ‘balanced’ helps us to in giving higher weights to the minority class and lower weights to the majority class.

Although passing value as ‘balanced’ gives good results in most cases but sometimes for extreme class imbalance, we can try giving weights manually. Later we will see how we can find the optimal value for the class weights in Python.

In our dataset:

Class 0 -> 1166

Class 1 -> 1173

Class 2 -> 12661

Totally 15000 train samples

So

Wj = n\_samples / (n\_classes \* n\_samples\_j)

W0 = 15000 / (3\*1166) = 4.288

W1 = 15000/ (3\*1173) = 4.263

W2 = 15000/ (3\*12661) = 0.395