**How to handle the Imbalance data?**

While working on my Data science Project my model predicted the good accuracy score for, but when I saw the confusion matrix it was surprising as my model was biased means predicting only one result compared to other so this how the imbalanced date attracts me.

So, one can see how this imbalanced data can affect your model as it can give wrong predictions.

**Let’s see first what this imbalance data means:**

If I am using supervise learning so we have some independent feature and one dependent feature that also called as target variable. In the target features if I have yes and no or 0 and 1. Now consider that out of 1000 total I have two types

1. 900 ‘yes’ and 100 ‘no’
2. 500 ‘yes’ and 500 ‘no’

Out of these second is look like balanced but first one is imbalanced data and the probability of predicting ‘yes’ will be increase compared to ‘no’. That’s why it’s very important to handle the imbalanced data.

So how to handle the imbalanced data?

There are two ways to handle the imbalanced dataset.

1. Down sampling/Under sampling
2. Up sampling/ over sampling

**Now let’s see what’s going on at the backend of these techniques**

**under sampling /Down sampling**

It means reducing the size of data.

let take an example of imbalance data we have 900 ‘yes’ and 100 ‘no’. In under sampling we are reducing the 900 of ‘yes’ to 100 ‘yes’.

Now I have 100 ‘yes’ and 100 ‘no’. So, this looks like balanced data. It means we have 50% ‘yes’ and 50% no.

**Up sampling/ over sampling**

It means increasing the size of lesser data.

It can be done by

1. Adding extra points over the lesser data or overlapping new data points on same points.
2. Creating new artificial points near the lesser data.
3. On the basis of class weight method

If we have 900 ‘yes’ and 100 ‘no’ then in up sampling we are increasing the 100 ‘no’ to 900 ‘no’ means we are overlapping the 800 new ‘no’ data points on previous 100 ‘no’ data points and it will become 900 ‘no’ data points.

Then I will obtain 900 ‘yes’ and 900 ‘no’ now it is 50% ‘yes’ and 50% no means we have got balanced data.

**Which technique should we use and how to use it?**

1. **Down sampling/under sampling**

In under sampling we are reducing the data points so there might be a chance that critical information remains unnoticed or we lose some precious information while reducing the data.

We can use this technique when data is in millions, otherwise its better to go with up sampling.

If the data is less, we should not use this technique.

We can use NearMiss under sampling from the imblearn library.

**Steps to use undersampler NearMiss**

pip install imblearn

from imblearn.under\_sampling import NearMiss

undersampling = NearMiss()

x\_res,y\_res = undersampling.fit\_resample(x\_train,y\_train)

from collections import Counter

print("The number of classes before fit {}".format(Counter(y\_train)))

print("The number of classes after fit {}".format(Counter(y\_res)))

1. **Over sampling /up sampling**

In over sampling there is no chance of unnoticing the critical information.

So, when the data is less, we should use over sampling methods.

There are two techniques for oversampling

Using SMOTETomek or use of random over sampler

**Steps for using Random over sampler**

from imblearn.over\_sampling import RandomOverSampler

os=RandomOverSampler()

x\_res,y\_res = os.fit\_resample(x\_train,y\_train)

from collections import Counter

print("The number of classes before fit {}".format(Counter(y\_train)))

print("The number of classes after fit {}".format(Counter(y\_res)))

**steps for using SMOTETomek**

from imblearn.combine import SMOTETomek

smk = SMOTETomek()

x\_res,y\_res = os.fit\_resample(x\_train,y\_train)