How to deal Imbalanced Data Set (Imbalance is only applicale in categorical label)

In [1]:

```
import pandas as pd
from sklearn.model_selection import train_test_split

import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
# you can imbalance the diabetes file yourself. open csv file directory and change most of

df=pd.read_csv('https://raw.githubusercontent.com/training-ml/Files/main/diabetes_up_down.c
df.head()
```

Out[2]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunctio
0	6	148	72	35	0	33.6	0.62
1	1	85	66	29	0	26.6	0.35
2	8	183	64	0	0	23.3	0.67
3	1	89	66	23	94	28.1	0.16
4	0	137	40	35	168	43.1	2.28

In [3]:

```
df['Outcome'].value_counts()
```

Out[3]:

```
0    720
1     48
Name: Outcome, dtype: int64
```

In [4]:

```
x=df.drop("Outcome",axis=1)
y=df.Outcome
```

In [5]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.75,random_state=41)
```

In [6]:

```
print('Training Outcome - \n',y_train.value_counts())
```

```
Training Outcome -
 0    544
1     32
Name: Outcome, dtype: int64
```

In [7]:

```
# you can also use counter module to count each class
from collections import Counter
Counter(y_train)
```

Out[7]:

```
Counter({1: 32, 0: 544})
```

Over Sampling using SMOTE (Synthetic Minority Oversampling Technique)

. This technique can be used when you have small data set

. This uses the euclidian distance method and find the closest data points to the minority class and create new minority data points

In [8]:

```
# if you face any issue with "from imblearn.over_sampling import SMOTE", then execute below
```

In [14]:

```
import six
import joblib
import sys
sys.modules['sklearn.extrnals.six']= six
sys.modules['sklearn.externals.joblib']=joblib
```

In [10]:

```
#pip install imblearn
```

In [11]:

```
#!pip install imblearn
```

In [16]:

```
# SMOTE picks the minority class nearest samples and create more samples

from imblearn.over_sampling import SMOTE
```

In [20]:

```
#sm=SMOTE()
```

In [23]:

```
ove_smp=SMOTE(0.75)
x_train_ns,y_train_ns=ove_smp.fit_resample(x_train,y_train)

print("The number of classes before fit{}".format(Counter(y_train)))
print("The number of classes afterfit{}".format(Counter(y_train_ns)))
```

```
The number of classes before fitCounter({0: 544, 1: 32})
The number of classes afterfitCounter({0: 544, 1: 408})
```

Downsampling using NearMiss

The technique can be used when you have large data set The uses the euclidian distance method and find the closest data points to the minority class and will eliminate the majority data points which are not close to the minority data points.

In [24]:

```
from imblearn.under_sampling import NearMiss
```

In [25]:

```
ds=NearMiss(0.75)
x_train_ns,y_train_ns=ds.fit_resample(x_train,y_train)

print("The number of classes before fit{}".format(Counter(y_train)))
print("The number of classes afterfit{}".format(Counter(y_train_ns)))
```

```
The number of classes before fitCounter({0: 544, 1: 32})
The number of classes afterfitCounter({0: 42, 1: 32})
```

In [ ]:

```

```