Handling imbalanced Dataset

July 11, 2024

When it contains unequal class distribution

e.g dataset with patients that are diabetic and non diabetic were diabetic are 1000 and non diabetic are 100 More number of data point for one particular class When working with imbalance dataset in python look at one of the class withh the less value then pick the one with high value data point and do a random selection of dataset that is close to the number of the one that is less and balance it before predicting in machine learning to give you a better prerdiction

[1]:
```
import pandas as pd
import numpy as ny
```

[2]:
```
#loading the dataset
credit = pd.read_csv('credit_data.csv')
```

[5]:
```
#first five rows
credit.head()
```

[5]:

	Time	V1	V2	V3	V4	V5	V6	V7	\
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	

	V8	V9	…	V21	V22	V23	V24	V25	\
0	0.098698	0.363787	…	-0.018307	0.277838	-0.110474	0.066928	0.128539	
1	0.085102	-0.255425	…	-0.225775	-0.638672	0.101288	-0.339846	0.167170	
2	0.247676	-1.514654	…	0.247998	0.771679	0.909412	-0.689281	-0.327642	
3	0.377436	-1.387024	…	-0.108300	0.005274	-0.190321	-1.175575	0.647376	
4	-0.270533	0.817739	…	-0.009431	0.798278	-0.137458	0.141267	-0.206010	

	V26	V27	V28	Amount	Class
0	-0.189115	0.133558	-0.021053	149.62	0
1	0.125895	-0.008983	0.014724	2.69	0
2	-0.139097	-0.055353	-0.059752	378.66	0
3	-0.221929	0.062723	0.061458	123.50	0
4	0.502292	0.219422	0.215153	69.99	0

```
[5 rows x 31 columns]
```

[8]:
```
credit.tail()
```

[8]:
```
            Time         V1         V2        V3        V4        V5  \
284802  172786.0 -11.881118  10.071785 -9.834783 -2.066656 -5.364473
284803  172787.0  -0.732789  -0.055080  2.035030 -0.738589  0.868229
284804  172788.0   1.919565  -0.301254 -3.249640 -0.557828  2.630515
284805  172788.0  -0.240440   0.530483  0.702510  0.689799 -0.377961
284806  172792.0  -0.533413  -0.189733  0.703337 -0.506271 -0.012546

               V6        V7        V8        V9  …       V21       V22  \
284802 -2.606837 -4.918215  7.305334  1.914428  …  0.213454  0.111864
284803  1.058415  0.024330  0.294869  0.584800  …  0.214205  0.924384
284804  3.031260 -0.296827  0.708417  0.432454  …  0.232045  0.578229
284805  0.623708 -0.686180  0.679145  0.392087  …  0.265245  0.800049
284806 -0.649617  1.577006 -0.414650  0.486180  …  0.261057  0.643078

             V23       V24       V25       V26       V27       V28  Amount  \
284802  1.014480 -0.509348  1.436807  0.250034  0.943651  0.823731    0.77
284803  0.012463 -1.016226 -0.606624 -0.395255  0.068472 -0.053527   24.79
284804 -0.037501  0.640134  0.265745 -0.087371  0.004455 -0.026561   67.88
284805 -0.163298  0.123205 -0.569159  0.546668  0.108821  0.104533   10.00
284806  0.376777  0.008797 -0.473649 -0.818267 -0.002415  0.013649  217.00

        Class
284802      0
284803      0
284804      0
284805      0
284806      0

[5 rows x 31 columns]
```

[10]:
```
# Determining the distribution of the two classes

credit['Class'].value_counts()
```

[10]:
```
Class
0    284315
1       492
Name: count, dtype: int64
```

0 = normal transaction 1 = fraudulent transaction

This is a Highly imbalanced dataset because we have -More number of data point for one particular class.

[13]:
```
#seperating the normal and fraudulent transactions

normal = credit[credit.Class == 0]

fraudulent = credit[credit.Class == 1]
```

[15]:
```
print(normal.shape)
print(fraudulent.shape)
```

```
(284315, 31)
(492, 31)
```

[ ]:
```
Implementing Undersampling to handle imbalance data
```

-Building a sample dataset containing similar distribution containing normal and fraudulent transaction

-fraudulent transactions == 498

-so you take a normal classification that is close to the number of the fraudulent number. take a random value of the normal value

[17]:
```
#n=492 is the random values I'm taking from the normal sample data

normal_sample = normal.sample(n=492)
```

[20]:
```
normal_sample.shape
```

[20]:
```
(492, 31)
```

[ ]:
```
Concatenating the two DataFrame
axis = 0 it means concatenating the data frame untop of each other
```

[21]:
```
new_dataset = pd.concat([normal_sample, fraudulent], axis = 0)
```

[23]:
```
new_dataset.head()
```

[23]:

	Time	V1	V2	V3	V4	V5	V6	\
5770	6155.0	-0.889223	1.164355	1.563641	0.980594	0.599625	-0.067060	
217877	141079.0	1.644255	-0.726884	-0.836590	1.247120	-0.216017	0.123339	
245292	152683.0	-0.609758	1.083061	-1.494478	-1.268186	1.133159	-0.805396	
115142	73763.0	-1.221469	0.916282	1.047121	0.249695	0.185015	-1.055604	
274291	165940.0	2.035648	-1.103028	-0.023296	-0.541400	-1.251403	0.233945	

	V7	V8	V9	…	V21	V22	V23	\
5770	0.714411	-0.251543	0.577944	…	-0.052967	0.084419	-0.415650	
217877	-0.164694	-0.104254	0.816753	…	0.288389	0.583782	-0.178243	
245292	1.141761	0.233349	-0.541029	…	0.316825	0.799008	-0.208133	
115142	0.719388	0.432021	-1.157914	…	-0.155634	-0.984461	-0.022736	
274291	-1.377625	0.179921	0.027736	…	-0.278814	-0.218307	0.314323	

```
              V24       V25       V26       V27       V28  Amount  Class
5770    -0.491558  0.602484 -0.158000 -0.216894  0.110248   35.00      0
217877 -0.984109  0.006609 -0.523698  0.023642 -0.003371  207.00      0
245292  0.310536 -0.322343  0.056838  0.142950  0.164443   29.99      0
115142  0.443228  0.514945 -0.751281 -0.197731 -0.150675   41.92      0
274291 -0.553916 -0.679902  0.500070  0.012925 -0.044575   25.90      0

[5 rows x 31 columns]
```

[24]:
```
new_dataset.tail()
```

[24]:
```
            Time        V1        V2        V3        V4        V5        V6  \
279863  169142.0 -1.927883  1.125653 -4.518331  1.749293 -1.566487 -2.010494
280143  169347.0  1.378559  1.289381 -5.004247  1.411850  0.442581 -1.326536
280149  169351.0 -0.676143  1.126366 -2.213700  0.468308 -1.120541 -0.003346
281144  169966.0 -3.113832  0.585864 -5.399730  1.817092 -0.840618 -2.943548
281674  170348.0  1.991976  0.158476 -2.583441  0.408670  1.151147 -0.096695

              V7        V8        V9  …       V21       V22       V23  \
279863 -0.882850  0.697211 -2.064945  …  0.778584 -0.319189  0.639419
280143 -1.413170  0.248525 -1.127396  …  0.370612  0.028234 -0.145640
280149 -2.234739  1.210158 -0.652250  …  0.751826  0.834108  0.190944
281144 -2.208002  1.058733 -1.632333  …  0.583276 -0.269209 -0.456108
281674  0.223050 -0.068384  0.577829  … -0.164350 -0.295135 -0.072173

              V24       V25       V26       V27       V28  Amount  Class
279863 -0.294885  0.537503  0.788395  0.292680  0.147968  390.00      1
280143 -0.081049  0.521875  0.739467  0.389152  0.186637    0.76      1
280149  0.032070 -0.739695  0.471111  0.385107  0.194361   77.89      1
281144 -0.183659 -0.328168  0.606116  0.884876 -0.253700  245.00      1
281674 -0.450261  0.313267 -0.289617  0.002988 -0.015309   42.53      1

[5 rows x 31 columns]
```

[26]:
```
new_dataset['Class'].value_counts()
```

[26]:
```
Class
0    492
1    492
Name: count, dtype: int64
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: