ML 3 - SMOTE(Synthetic Minority Oversampling Techniques) By Virat Tiwari

November 7, 2023

1 SMOTE (Synthetic Minority Oversampling Techniques)

SMOTE - Synthetic Minority Oversampling Technique is a technique used in machine learning to address imbalanced datasets where the minority class has significantly fewer instances than the majority class . SMOTE involves generating the synthetic instances of the minority class by interpolating between existing instance . In very simple terms it handles the imbalanced datasets.

Interpolating - In iterpolation technique we generate the data between two existance data points

[5]:
```
# Scikit-Learn is a free machine learning library for Python. It supports both␣
 ↪supervised and unsupervised machine learning, providing diverse algorithms␣
 ↪for classification, regression, clustering, and dimensionality reduction.␣
 ↪The library is built using many libraries you may already be familiar with,␣
 ↪such as NumPy and SciPy. It also plays well with other libraries, such as␣
 ↪Pandas and Seaborn.

# Here we use or import datasets from skelarn

# make_classification - We make specific classification with the help of␣
 ↪"make_classification function"

from sklearn.datasets import make_classification
```

[6]:
```
# x is independent feature
# y is dependent feature

x,y=make_classification(n_samples=1000,n_features=2,n_redundant=0,n_clusters_per_class=1,weigh
 ↪90],random_state=1)
```

[7]:
```
# x - This is independent variable or input

x
```

[7]:
```
array([[ 1.53682958, -1.39869399],
       [ 1.55110839,  1.81032905],
       [ 1.29361936,  1.01094607],
       …,
```

```
       [-0.55662536, -0.15983725],
       [ 1.00499902,  0.93628981],
       [ 1.46210987,  1.14497791]])
```

[8]:
```
# y - This is dependent feature of output

y
```

[8]:
```
array([1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

[9]:
```
# Here we CONVERT x and y in DATAFRAME using pandas

# f1 and f2 are input data or independent data

# target is output or dependent data

import pandas as pd
df1=pd.DataFrame(x,columns=["f1","f2"])
df2=pd.DataFrame(y,columns=["target"])
final_df=pd.concat([df1,df2],axis=1)
```

[10]:
```
final_df.head()
```

[10]:

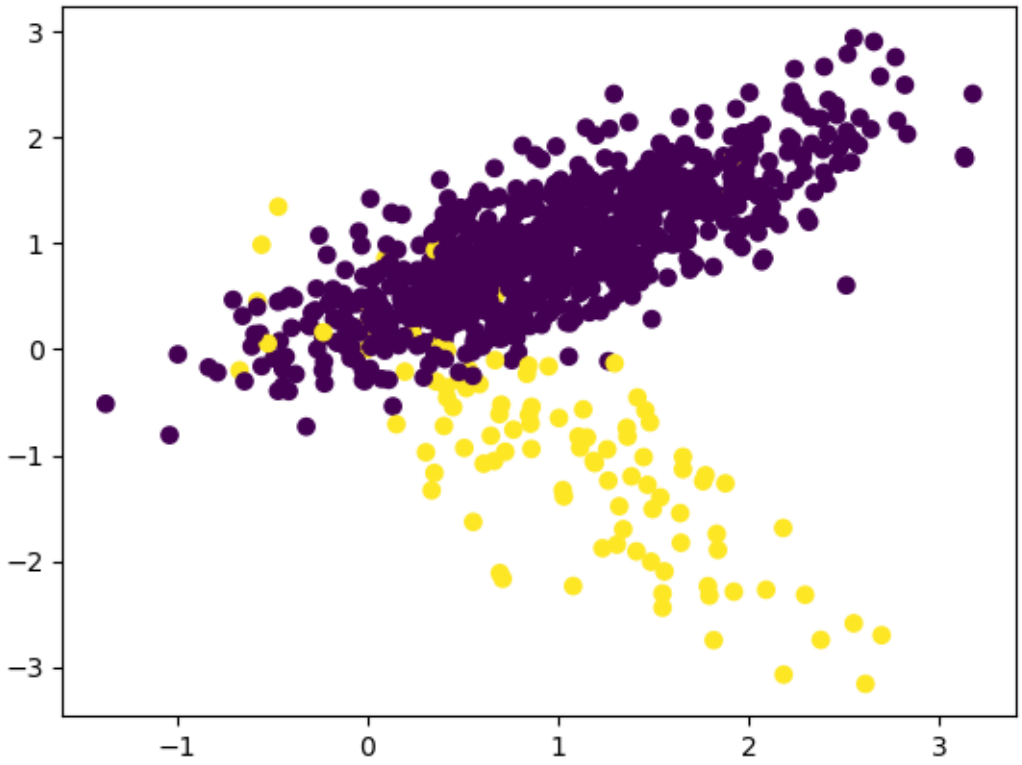
	f1	f2	target
0	1.536830	-1.398694	1
1	1.551108	1.810329	0
2	1.293619	1.010946	0
3	1.119889	1.632518	0
4	1.042356	1.121529	0

[11]:
```
# o comes 894 times in dataset
# 1 comes 106 times in a dataset
# This is purely imbalanced dataset

final_df["target"].value_counts()
```

[11]:
```
0    894
1    106
Name: target, dtype: int64
```

[12]:
```
import matplotlib.pyplot as plt
plt.scatter(final_df["f1"],final_df["f2"],c=final_df["target"])

# Note - In this graph we have seen that "YELLOW" is minority datapoints which␣
 ↪store in "x" and "PURPLE" is majority datapoints which store in "y"
```

[12]:
```
<matplotlib.collections.PathCollection at 0x7f399ba5b3d0>
```



[13]:
```
# FOR PERFORMING THE SMOTE" - WE HAVE TO IMPORT IMBLEARN

pip install imblearn
```

```
Requirement already satisfied: imblearn in /opt/conda/lib/python3.10/site-
packages (0.0)
Requirement already satisfied: imbalanced-learn in
/opt/conda/lib/python3.10/site-packages (from imblearn) (0.11.0)
Requirement already satisfied: joblib>=1.1.1 in /opt/conda/lib/python3.10/site-
packages (from imbalanced-learn->imblearn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/opt/conda/lib/python3.10/site-packages (from imbalanced-learn->imblearn)
(3.1.0)
Requirement already satisfied: scikit-learn>=1.0.2 in
/opt/conda/lib/python3.10/site-packages (from imbalanced-learn->imblearn)
(1.2.0)
Requirement already satisfied: numpy>=1.17.3 in /opt/conda/lib/python3.10/site-
packages (from imbalanced-learn->imblearn) (1.23.5)
Requirement already satisfied: scipy>=1.5.0 in /opt/conda/lib/python3.10/site-
packages (from imbalanced-learn->imblearn) (1.9.3)
Note: you may need to restart the kernel to use updated packages.
```

[14]:
```
from imblearn.over_sampling import SMOTE
```

[15]:
```
# HERE WE TRANSFORM IMBALACED DADASET INTO BALANCED DATASET USING SMOTE␣
↪TECHNIQUE

# Tranform then dataset

oversample=SMOTE()
x,y=oversample.fit_resample(final_df[["f1","f2"]],final_df["target"])
```

[16]:
```
# . shape ( ) function gives the no of datapoints from the specif feature of␣
↪dataset or section

x.shape
```

[16]:
```
(1788, 2)
```

[17]:
```
y.shape
```

[17]:
```
(1788,)
```

[21]:
```
len(y[y==0])
```

[21]:
```
894
```

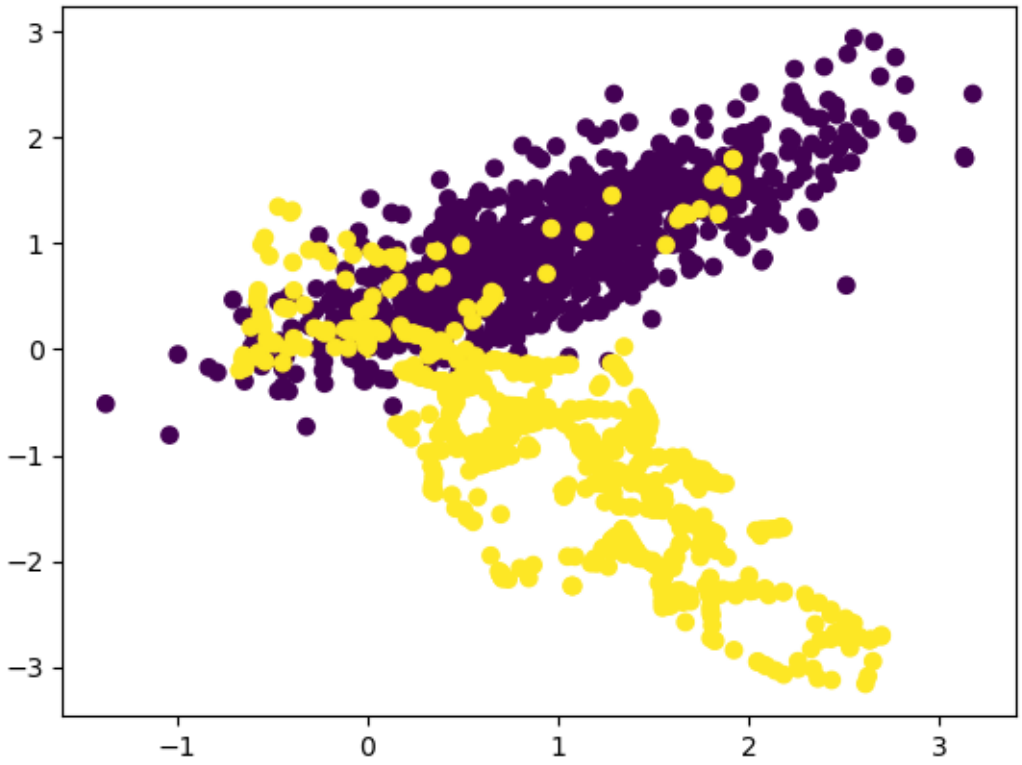
[22]:
```
len(x[x==0])
```

[22]:
```
1788
```

[23]:
```
# For visualizing the new dadaset we have to perform these operations

import pandas as pd
df1=pd.DataFrame(x,columns=["f1","f2"])
df2=pd.DataFrame(y,columns=["target"])
oversample_df=pd.concat([df1,df2],axis=1)
```

[24]:
```
plt.scatter(oversample_df["f1"],oversample_df["f2"],c=oversample_df["target"])

# Note - In this graph datapoints are equal it mean our x and y both have equal␣
↪datapoints and this is how we handle the imbalanced dataset
```

[24]:
```
<matplotlib.collections.PathCollection at 0x7f39af7c19f0>
```



THANK YOU SO MUCH !!

YOURS VIRAT TIWARI :)