Resampling Techniques

Tushar B. Kute,
http://tusharkute.com

Imbalanced Data

- It is a scenario where the number of observations belonging to one class is significantly lower than those belonging to the other classes.
- This problem is predominant in scenarios where anomaly detection is crucial like electricity pilferage, fraudulent transactions in banks, identification of rare diseases, etc. In this situation, the predictive model developed using conventional machine learning algorithms could be biased and inaccurate.
- This happens because Machine Learning Algorithms are usually designed to improve accuracy by reducing the error. Thus, they do not take into account the class distribution / proportion or balance of classes.

Why to work on imbalanced data?

- Machine Learning algorithms tend to produce unsatisfactory classifiers when faced with imbalanced datasets.
- For any imbalanced data set, if the event to be predicted belongs to the minority class and the event rate is less than 5%, it is usually referred to as a rare event.

Example:

- Let’s understand this with the help of an example.
- Ex: In an utilities fraud detection data set you have the following data:
  - Total Observations = 1000
  - Fraudulent Observations = 20
  - Non Fraudulent Observations = 980
  - Event Rate= 2 %
- The main question faced during data analysis is – How to get a balanced dataset by getting a decent number of samples for these anomalies given the rare occurrence for some them?

Challenges in ML Algorithms

- The conventional model evaluation methods do not accurately measure model performance when faced with imbalanced datasets.
- Standard classifier algorithms like Decision Tree and Logistic Regression have a bias towards classes which have number of instances.
- They tend to only predict the majority class data. The features of the minority class are treated as noise and are often ignored.
- Thus, there is a high probability of misclassification of the minority class as compared to the majority class.

Few more examples

- Apart from fraudulent transactions, other examples of a common business problem with imbalanced dataset are:
  - Datasets to identify customer churn where a vast majority of customers will continue using the service. Specifically, Telecommunication companies where Churn Rate is lower than 2 %.
  - Data sets to identify rare diseases in medical diagnostics etc.
  - Natural Disaster like Earthquakes

Resampling Techniques

- Dealing with imbalanced datasets entails strategies such as improving classification algorithms or balancing classes in the training data (data preprocessing) before providing the data as input to the machine learning algorithm. The later technique is preferred as it has wider application.
- The main objective of balancing classes is to either increasing the frequency of the minority class or decreasing the frequency of the majority class. This is done in order to obtain approximately the same number of instances for both the classes.

Resampling Techniques

- Random Under Sampling
- Random Over Sampling
- SMOTE

Random Under Sampling

- Random Undersampling aims to balance class distribution by randomly eliminating majority class examples. This is done until the majority and minority class instances are balanced out.
  - Total Observations = 1000
  - Fraudulent Observations =20
  - Non Fraudulent Observations = 980
  - Event Rate= 2 %
- In this case we are taking 10 % samples without replacement from Non Fraud instances. And combining them with Fraud instances.
  - Non Fraudulent Observations after random under sampling = 10 % of 980 =98
  - Total Observations after combining them with Fraudulent observations = 20+98=118
  - Event Rate for the new dataset after under sampling = 20/118 = 17%

Random Under Sampling

- Advantages
  - It can help improve run time and storage problems by reducing the number of training data samples when the training data set is huge.
- Disadvantages
  - It can discard potentially useful information which could be important for building rule classifiers.
  - The sample chosen by random under sampling may be a biased sample. And it will not be an accurate representative of the population. Thereby, resulting in inaccurate results with the actual test data set.

Random Over Sampling

- Over-Sampling increases the number of instances in the minority class by randomly replicating them in order to present a higher representation of the minority class in the sample.
  - Total Observations = 1000
  - Fraudulent Observations =20
  - Non Fraudulent Observations = 980
  - Event Rate= 2 %
- In this case we are replicating 20 fraud observations 20 times.
  - Non Fraudulent Observations =980
  - Fraudulent Observations after replicating the minority class observations= 400
  - Total Observations in the new data set after oversampling=1380
  - Event Rate for the new data set after under sampling= 400/1380 = 29 %

Random Over Sampling

- Advantages
  - Unlike under sampling this method leads to no information loss.
  - Outperforms under sampling
- Disadvantages
  - It increases the likelihood of overfitting since it replicates the minority class events.

SMOTE

- SMOTE (Synthetic Minority Oversampling Technique) synthesises new minority instances between existing minority instances.
- It randomly picks up the minority class and calculates the K-nearest neighbour for that particular point.
- Finally, the synthetic points are added between the neighbours and the chosen spot.

Cluster based over sampling

- K means clustering algorithm is independently applied to both the class instances such as to identify clusters in the datasets. All clusters are oversampled such that clusters of the same class have the same size.
- For Example –
  - Total Observations : 100
  - Positive Dataset : 90
  - Negative Dataset : 10
  - Event Rate : 2%
- Majority Class Cluster:
  - Cluster 1: 20 Observations
  - Cluster 2: 30 Observations
  - Cluster 3: 12 Observations
  - Cluster 4: 18 Observations
  - Cluster 5: 10 Observations
- Minority Class Cluster:
  - Cluster 1: 8 Observations
  - Cluster 2: 12 Observations

After over sampling

- After oversampling all clusters of the same class have the same number of observations.
- Majority Class Cluster:
  - Cluster 1: 20 Observations
  - Cluster 2: 20 Observations
  - Cluster 3: 20 Observations
  - Cluster 4: 20 Observations
  - Cluster 5: 20 Observations
- Minority Class Cluster:
  - Cluster 1: 15 Observations
  - Cluster 2: 15 Observations

Installation needed

- Imbalanced Learning Library
  - `sudo pip3 install imblearn -U`
- Data Importing
  - `sudo pip3 install pandas -U`
- Visualization
  - `sudo pip3 install seaborn -U`
- Machine Learning Packages
  - `sudo pip3 install scikit-learn -U`

Dataset

- Credit Card Fraud Detection Dataset
  - https://www.kaggle.com/mlg-ulb/creditcardfraud

Data Reading and Analysis

```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from imblearn.under_sampling import RandomUnderSampler, TomekLinks
from imblearn.over_sampling import RandomOverSampler, SMOTE
```

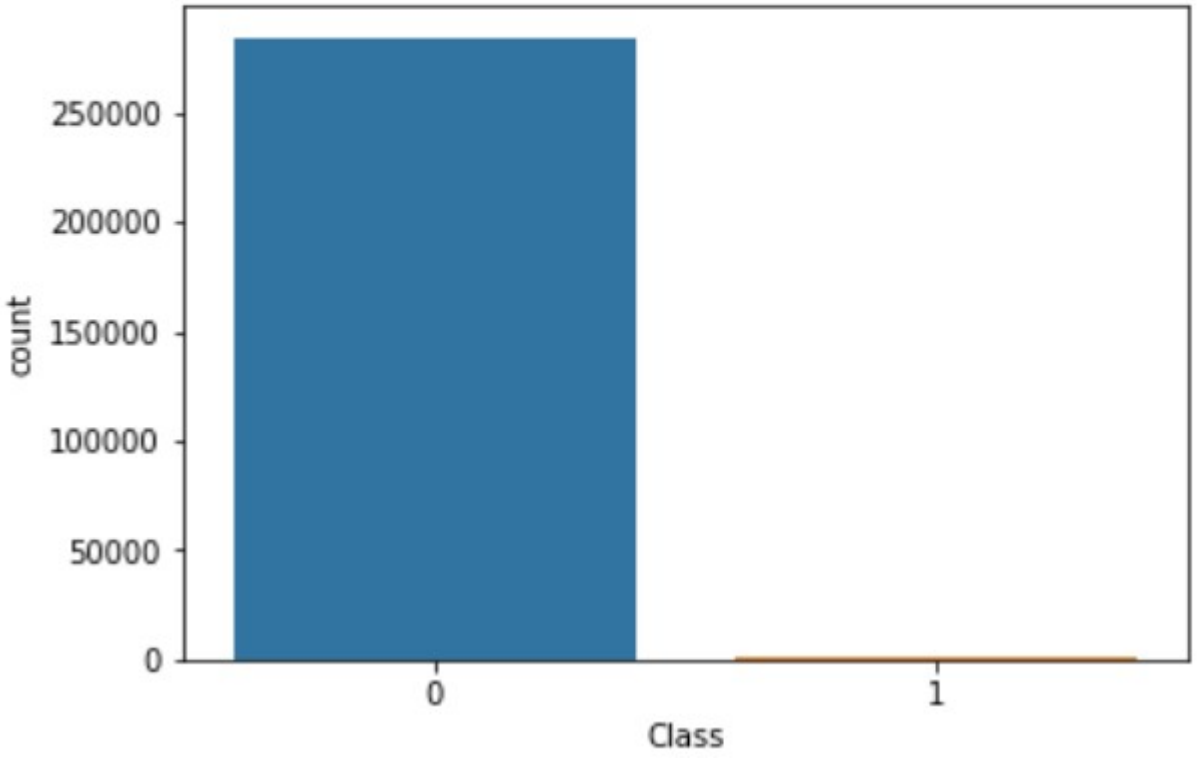
```
dataset = pd.read_csv('creditcard.csv')
print("The Number of Samples in the dataset: ", len(dataset))
print('Class 0        :', round(dataset['Class'].value_counts()[0]
                      /len(dataset) * 100, 2), '% of the dataset')

print('Class 1(Fraud) :', round(dataset['Class'].value_counts()[1]
                      /len(dataset) * 100, 2), '% of the dataset')
```

```
The Number of Samples in the dataset:  284807
Class 0        : 99.83 % of the dataset
Class 1(Fraud) : 0.17 % of the dataset
```

Count plot

```
sns.countplot(dataset['Class'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3464bc2ba8>
```



Under Sampling

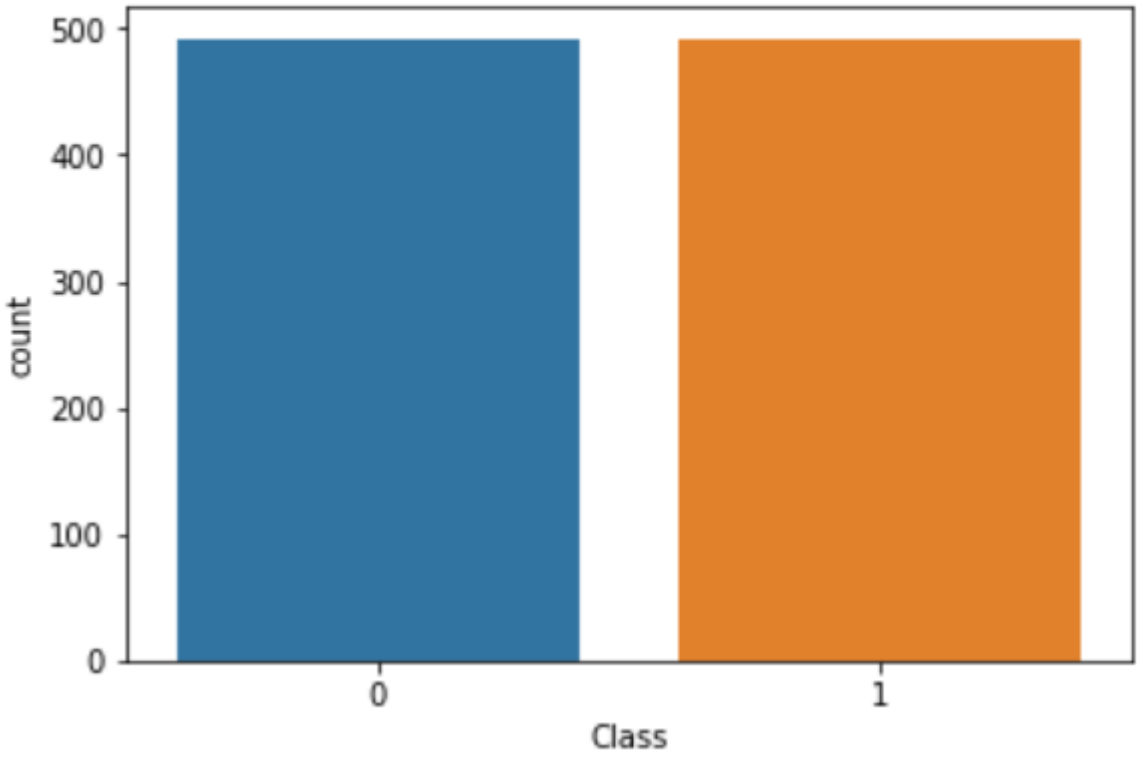
```
X_data = dataset.iloc[:, :-1]
Y_data = dataset.iloc[:, -1:]

rus = RandomUnderSampler(random_state = 0)
X_res, y_res = rus.fit_resample(X_data, Y_data)

X_res = pd.DataFrame(X_res)
Y_res = pd.DataFrame(y_res)

print("After Under Sampling: Major Class Total Samples:", len(Y_res))
print('Class 0        :', round(Y_res['Class'].value_counts()[0]
                      /len(Y_res) * 100, 2), '% of the dataset')

print('Class 1(Fraud) :', round(Y_res['Class'].value_counts()[1]
                      /len(Y_res) * 100, 2), '% of the dataset')
```

```
After Under Sampling Of Major Class Total Samples are : 984
Class 0        : 50.0 % of the dataset
Class 1(Fraud) : 50.0 % of the dataset
```

After Under Sampling

```
sns.countplot(Y_res['Class'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f34621cf358>
```



Over Sampling

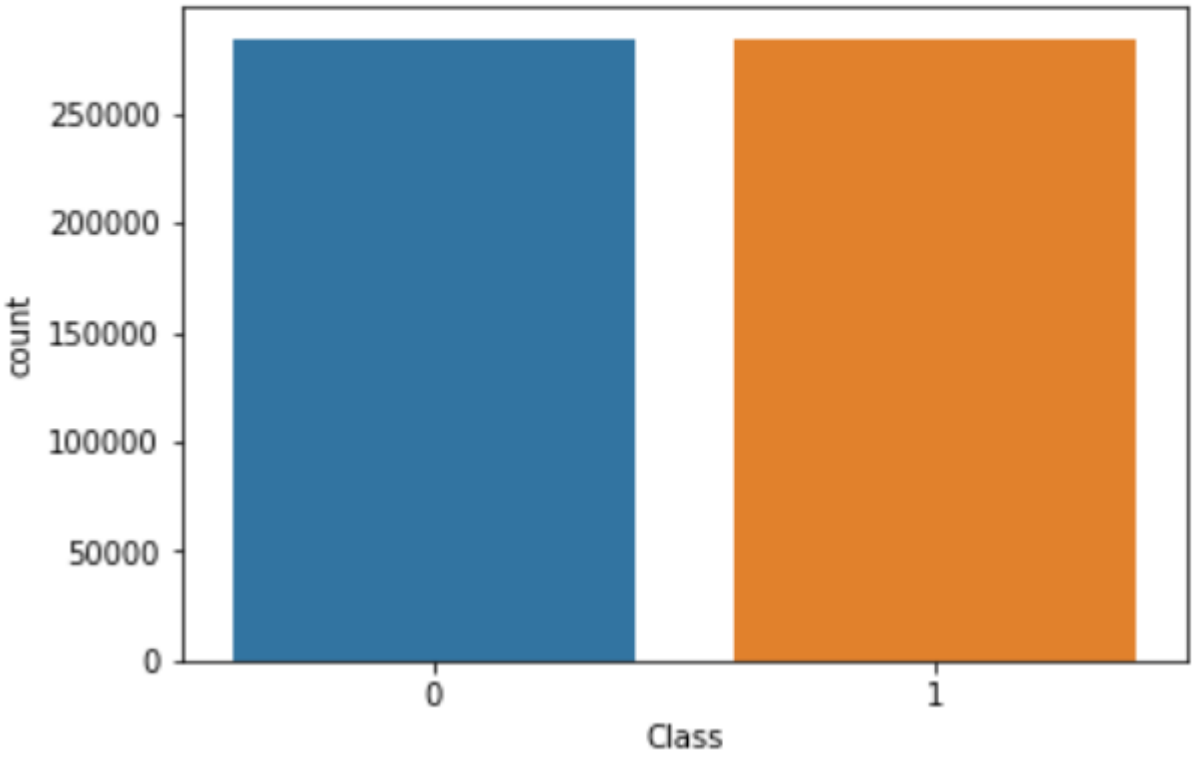
```
X_data = dataset.iloc[:, :-1]
Y_data = dataset.iloc[:, -1:]

rus = RandomOverSampler(random_state = 0)
X_res, y_res = rus.fit_resample(X_data, Y_data)

X_res = pd.DataFrame(X_res)
Y_res = pd.DataFrame(y_res)

print("After Over Sampling: Major Class Total Samples:", len(Y_res))
print('Class 0        :', round(Y_res['Class'].value_counts()[0]
                      /len(Y_res) * 100, 2), '% of the dataset')

print('Class 1(Fraud) :', round(Y_res['Class'].value_counts()[1]
                      /len(Y_res) * 100, 2), '% of the dataset')
```

```
After Under Sampling Of Major Class Total Samples are : 568630
Class 0        : 50.0 % of the dataset
Class 1(Fraud) : 50.0 % of the dataset
```

After Over Sampling

```
sns.countplot(Y_res['Class'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f346196d898>
```



SMOTE

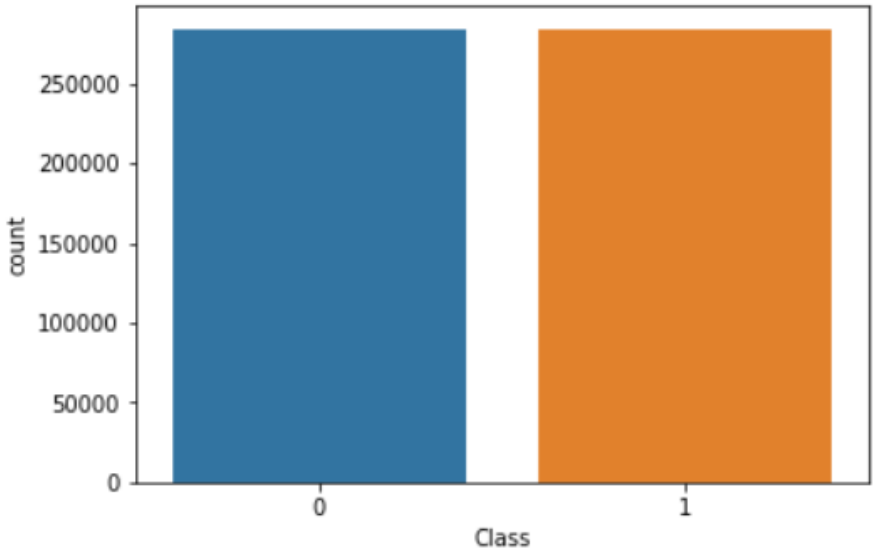
```
X_data = dataset.iloc[:, :-1]
Y_data = dataset.iloc[:, -1:]

rus = SMOTE(random_state = 0)
X_res, y_res = rus.fit_resample(X_data, Y_data)

X_res = pd.DataFrame(X_res)
Y_res = pd.DataFrame(y_res)

print("After SMOTE Of Major Class Total Samples are :", len(Y_res))
print('Class 0        :', round(Y_res['Class'].value_counts()[0]
                      /len(Y_res) * 100, 2), '% of the dataset')

print('Class 1(Fraud) :', round(Y_res['Class'].value_counts()[1]
                      /len(Y_res) * 100, 2), '% of the dataset')
```

```
After SMOTE Of Major Class Total Samples are : 568630
Class 0        : 50.0 % of the dataset
Class 1(Fraud) : 50.0 % of the dataset
```

After SMOTE

```
sns.countplot(Y_res['Class'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f34619074a8>
```




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

```
1    492
0    492
Name: Class, dtype: int64
```

Clustering

```
from imblearn.under_sampling import ClusterCentroids
```

```
X_data = dataset.iloc[:, :-1]
Y_data = dataset.iloc[:, -1:]

rus = ClusterCentroids(random_state = 0)
X_res, y_res = rus.fit_resample(X_data, Y_data)

X_res = pd.DataFrame(X_res)
Y_res = pd.DataFrame(y_res)

print("After Cluster Major Class Total Samples are :", len(Y_res))
print('Class 0        :', round(Y_res['Class'].value_counts()[0]
                      /len(Y_res) * 100, 2), '% of the dataset')

print('Class 1(Fraud) :', round(Y_res['Class'].value_counts()[1]
                      /len(Y_res) * 100, 2), '% of the dataset')
```

Useful resources

- https://imbalanced-learn.readthedocs.io/
- www.geeksforthegeeks.org
- www.scikit-learn.org
- www.towardsdatascience.com
- www.medium.com
- www.analyticsvidhya.com
- www.kaggle.com
- www.stephacking.com
- www.github.com

Thank you

This presentation is created using LibreOffice Impress 5.1.6.2, can be used freely as per GNU General Public License

@mitu_skillologies

/mITuSkillologies

@mitu_group

/company/mitu-skillologies

MITUSkillologies

Web Resources
https://mitu.co.in
http://tusharkute.com

contact@mitu.co.in

tushar@tusharkute.com