Published in Towards Data Science

Satyam Kumar Follow
Nov 12, 2020 · 5 min read · ✦ · Listen

# 7 Over Sampling techniques to handle Imbalanced Data

Deep dive analysis of various oversampling techniques

Image by LTD EHU from Pixabay

Modeling imbalanced data is the major challenge that we face when we train a

Some of the famous examples of imbalanced class problems are:

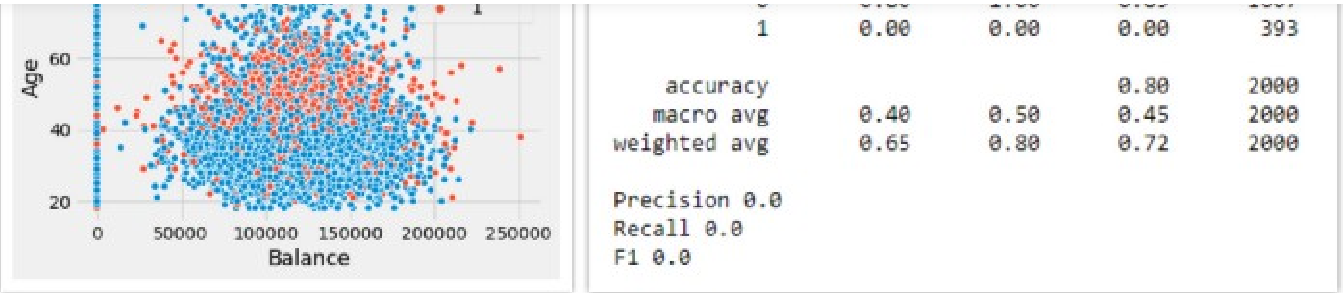
1. Credit Card Fraud Detection
2. Disease diagnosis
3. Spam detection, and many more

The imbalance of the dataset needs to be handled before training a model. There are various techniques to handle class balance, some of them being Oversampling, Undersampling, or a combination of both. This article will cover a deep dive explanation of 7 techniques of oversampling:

1. **Random Over Sampling**
2. **Smote**
3. **BorderLine Smote**
4. **KMeans Smote**
5. **SVM Smote**
6. **ADASYN**
7. **Smote-NC**

> *For the evaluation of different oversampling models, we are using the Churn modeling dataset from Kaggle.*

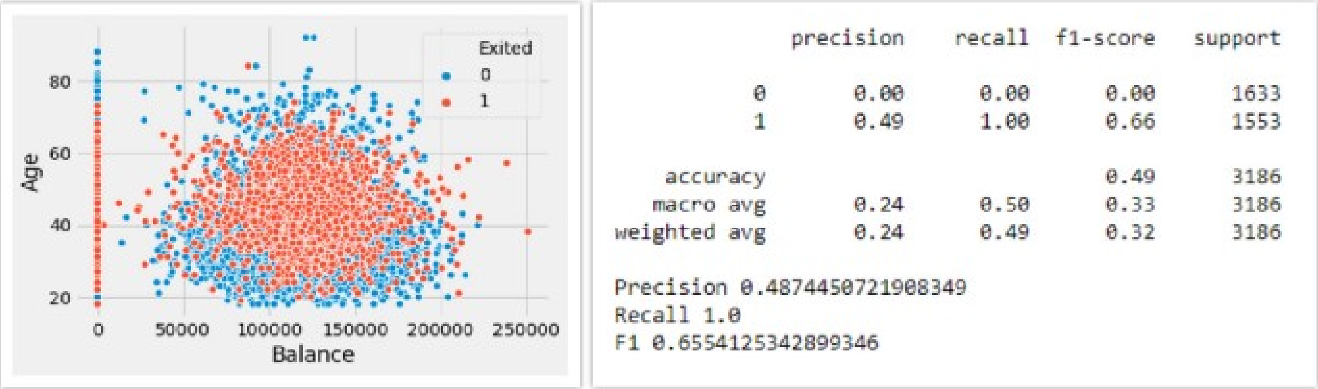
**Performace of the Logistic Regression model without using any oversampling or undersampling technique.**



```
           1       0.00      0.00      0.00       393

    accuracy                           0.80      2000
   macro avg       0.40      0.50      0.45      2000
weighted avg       0.65      0.80      0.72      2000

Precision 0.0
Recall 0.0
F1 0.0
```

## 1. Random Over Sampling:

Random oversampling is the simplest oversampling technique to balance the imbalanced nature of the dataset. It balances the data by replicating the minority class samples. This does not cause any loss of information, but the dataset is prone to overfitting as the same information is copied.



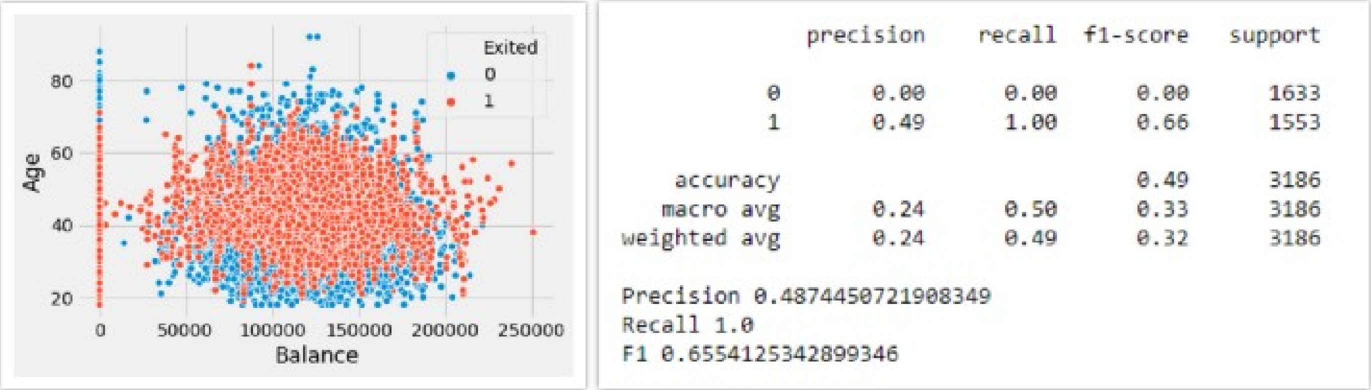
```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00      1633
           1       0.49      1.00      0.66      1553

    accuracy                           0.49      3186
   macro avg       0.24      0.50      0.33      3186
weighted avg       0.24      0.49      0.32      3186

Precision 0.4874450721908349
Recall 1.0
F1 0.6554125342899346
```

(Image by Author), **Left:** Scatter plot after Random Oversampling, **Right:** Performance of model after Random Oversampling

## 2. SMOTE:

In the case of random oversampling, it was prone to overfitting as the minority class samples are replicated, here SMOTE comes into the picture. SMOTE stands for Synthetic Minority Oversampling Technique. It creates new synthetic samples to balance the dataset.

- Identify the feature vector and its nearest neighbor
- Compute the distance between the two sample points
- Multiply the distance with a random number between 0 and 1.
- Identify a new point on the line segment at the computed distance.
- Repeat the process for identified feature vectors.



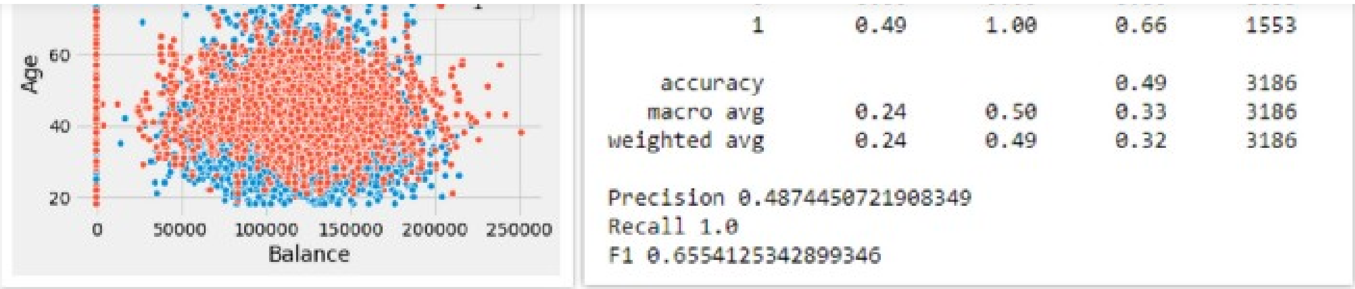
```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00      1633
           1       0.49      1.00      0.66      1553

    accuracy                           0.49      3186
   macro avg       0.24      0.50      0.33      3186
weighted avg       0.24      0.49      0.32      3186

Precision 0.4874450721908349
Recall 1.0
F1 0.6554125342899346
```

(Image by Author), **Left:** Scatter plot after SMOTE, **Right:** Performance of model after SMOTE

### 3. Borderline Smote:

Due to the presence of some minority points or outliers within the region of majority class points, bridges of minority class points are created. This is a problem in the case of Smote and is solved using Borderline Smote.

In the Borderline Smote technique, only the minority examples near the borderline are over-sampled. It classifier the minority class points into noise points, border points. Noise points are minority class points that have most of the points as majority points in its neighbor, and border points have both majority and minority class points in its neighbor. Borderline Smote algorithm tries to create synthetic points using only these



```
           1       0.49      1.00      0.66      1553

    accuracy                           0.49      3186
   macro avg       0.24      0.50      0.33      3186
weighted avg       0.24      0.49      0.32      3186

Precision 0.4874450721908349
Recall 1.0
F1 0.6554125342899346
```

(Image by Author), **Left:** Scatter plot after Borderline SMOTE, **Right:** Performance of model after Borderline SMOTE

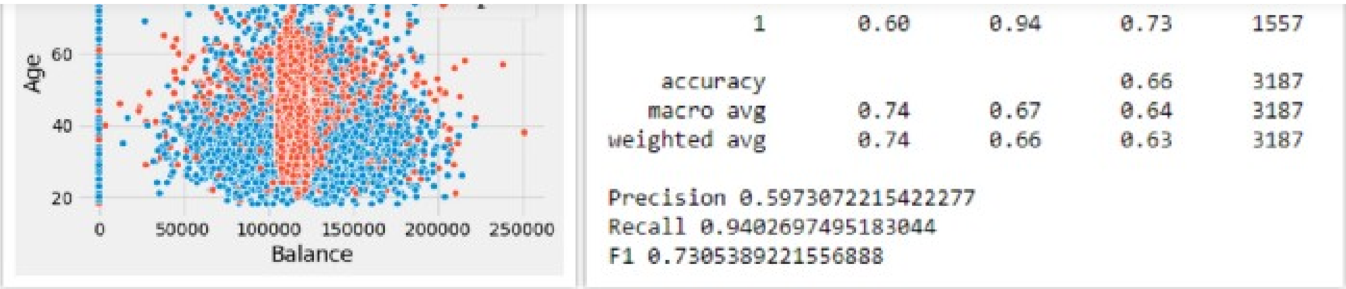
## 4. KMeans Smote:

K-Means SMOTE is an oversampling method for class-imbalanced data. It aids classification by generating minority class samples in safe and crucial areas of the input space. The method avoids the generation of noise and effectively overcomes imbalances between and within classes.

K-Means SMOTE works in five steps:

1. Cluster the entire data using the k-means clustering algorithm.
2. Select clusters that have a high number of minority class samples
3. Assign more synthetic samples to clusters where minority class samples are sparsely distributed.

Here each filtered cluster is oversampled using SMOTE.



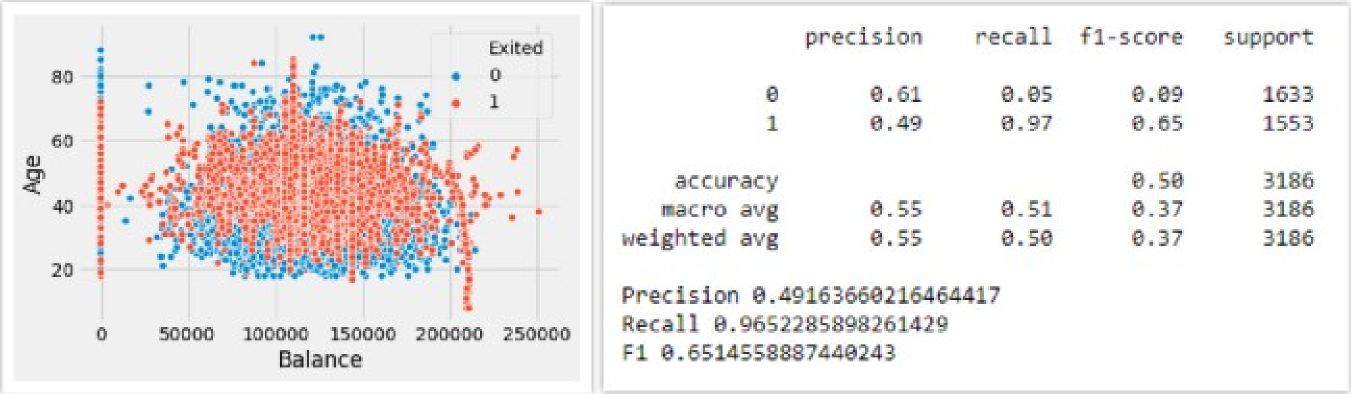
```
           1       0.60      0.94      0.73      1557

    accuracy                           0.66      3187
   macro avg       0.74      0.67      0.64      3187
weighted avg       0.74      0.66      0.63      3187

Precision 0.5973072215422277
Recall 0.9402697495183044
F1 0.7305389221556888
```

(Image by Author), **Left:** Scatter plot after KMeans SMOTE, **Right:** Performance of model after KMeans SMOTE

## 5. SVM Smote:

Another variation of Borderline-SMOTE is Borderline-SMOTE SVM, or we could just call it SVM-SMOTE. This technique incorporates the SVM algorithm to identify the misclassification points.

In the SVM-SMOTE, the borderline area is approximated by the support vectors after training SVMs classifier on the original training set. Synthetic data is then randomly created along the lines joining each minority class support vector with a number of its nearest neighbors.

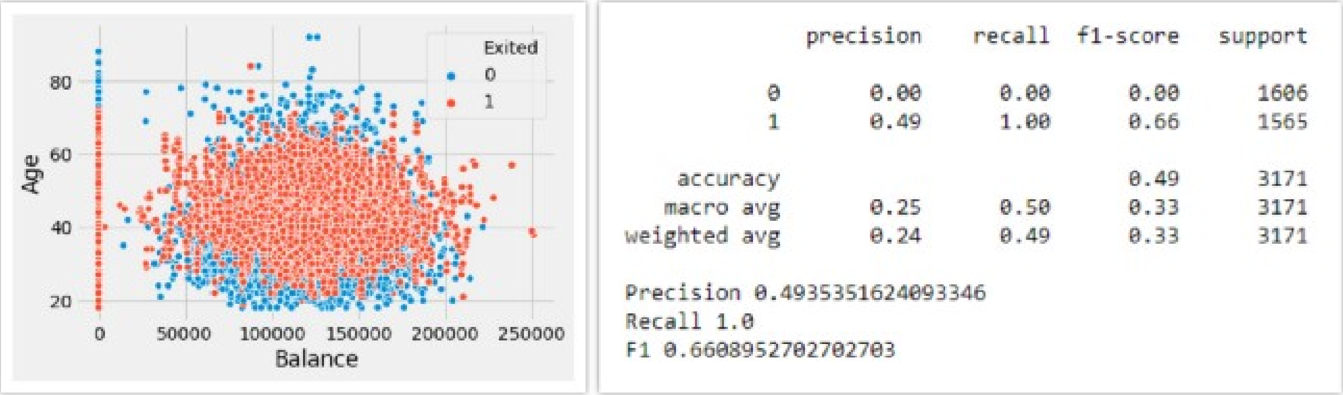


```
              precision    recall  f1-score   support

           0       0.61      0.05      0.09      1633
           1       0.49      0.97      0.65      1553

    accuracy                           0.50      3186
   macro avg       0.55      0.51      0.37      3186
weighted avg       0.55      0.50      0.37      3186

Precision 0.49163660216464417
Recall 0.9652285898261429
F1 0.6514558887440243
```

(Image by Author), **Left:** Scatter plot after SVM SMOTE, **Right:** Performance of model after SVM SMOTE points. This problem is solved by the ADASYN algorithm, as it **creates synthetic data according to the data density.**

The synthetic data generation is inversely proportional to the density of the minority class. A comparatively larger number of synthetic data is created in regions of a low density of minority class than higher density regions.

In other terms, in the less dense area of the minority class, the synthetic data are created more.



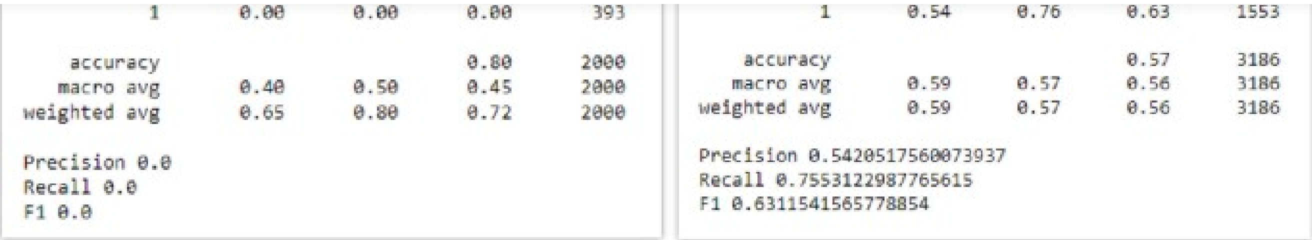
```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00      1606
           1       0.49      1.00      0.66      1565

    accuracy                           0.49      3171
   macro avg       0.25      0.50      0.33      3171
weighted avg       0.24      0.49      0.33      3171

Precision 0.4935351624093346
Recall 1.0
F1 0.6608952702702703
```

(Image by Author), **Left:** Scatter plot after ADASYN, **Right:** Performance of model after ADASYN

## 7. Smote-NC:

Smote oversampling technique only works for the dataset with all continuous features. For a dataset with categorical features, we have a variation of Smote, which is Smote-NC (Nominal and Continuous).

Smote can also be used for data with categorical features, by one-hot encoding but it may result in an increase in dimensionality. Label Encoding can also be used to convert categorical to numerical, but after smote it may result in unnecessary information. This is why we need to use SMOTE-NC when we have cases of mixed data. Smote-NC can be used by denoting the features that are categorical, and Smote would



```
           1       0.00      0.00      0.00       393

    accuracy                           0.80      2000
   macro avg       0.40      0.50      0.45      2000
weighted avg       0.65      0.80      0.72      2000

Precision 0.0
Recall 0.0
F1 0.0
```

```
           1       0.54      0.76      0.63      1553

    accuracy                           0.57      3186
   macro avg       0.59      0.57      0.56      3186
weighted avg       0.59      0.57      0.56      3186

Precision 0.5420517560073937
Recall 0.7553122987765615
F1 0.6311541565778854
```

(Image by Author), **Left:** Performance of model before SMOTE-NC, **Right:** Performance of model after SMOTE-NC

## Implementation:

(Code Implementation by Author)

## Conclusion:

Modeling an imbalanced dataset is the major challenge that we face while training a model, using various oversampling techniques discussed above the performance of the model can be improved. Also in this article, we have discussed SMOTE-NC, which is a variation of SMOTE, that can handle categorical features.

Model performance of an Imbalanced dataset can also be improved by using various undersampling techniques such as Random Undersampling, TomekLinks, etc, and a combination of oversampling and undersampling techniques such as SMOTEENN, SMOTETomek, etc.

## References:

[1] Imblearn documentation: https://imbalanced-learn.readthedocs.io/en/stable/api.html#module-imblearn.over_sampling

[2] https://pypi.org/project/kmeans-smote/

Thank You for Reading