Learning from imbalanced data

Ido Zehori

Big Data Analytics #4

Hello!

I am Ido Zehori

Data scientist @ 

You can find me at:

@IZehori

idozehori.com

@IdoZehori

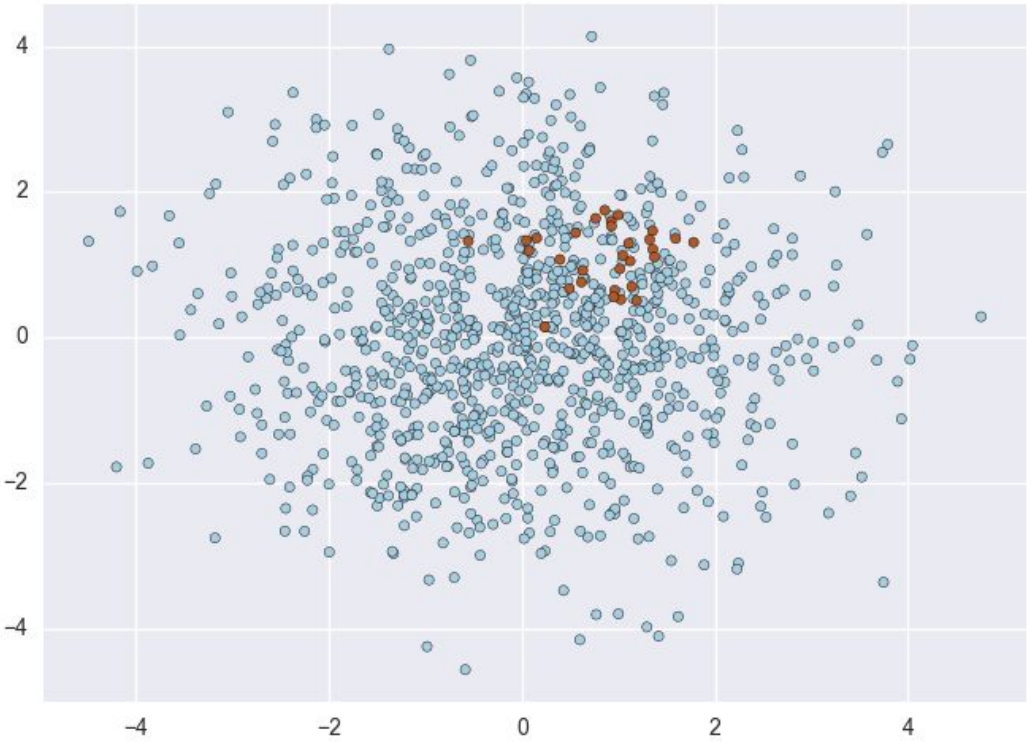
Agenda

- What is imbalanced data
- The accuracy paradox
- Alternative performance metrics
- Ratio parameter
- Resampling methods
- Ensemble methods

1.
What is Imbalanced Data?

What is Imbalanced Data?

- Cost of misclassification is higher on minority.
- Distribution is not uniform among the classes.

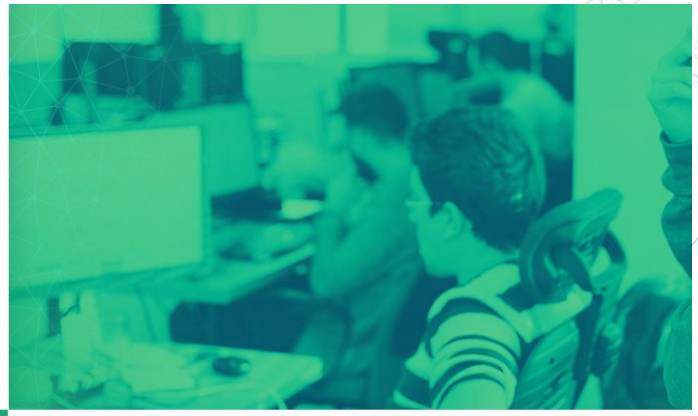


Where can we find Imbalanced Data?

Bigabid
SCIENTIFIC ADVERTISING

WHAT WE DO

BigaBid is the developer of the **BigaBid Proprietary Autonomous Platform**, which is compiled of a Demand Side Platform (**DSP**), a Data Management Platform (**DMP**) and the **BigaBid Brain** (a decision making and optimization set of algorithms, powered by a robust set of in-house developed machine learning engines).



We use our technology to decipher the habits and behavior of online users across the mobile ecosystem, and in turn, match the right user with the right offer, at the right time, with an emphasis on acquiring high Lifetime-Value customers for our advertiser clients.

Our aim is to ensure that users who view our ads are happy with offers that they are exposed to, and to satisfy our advertiser client's needs for high value customers.

The stack

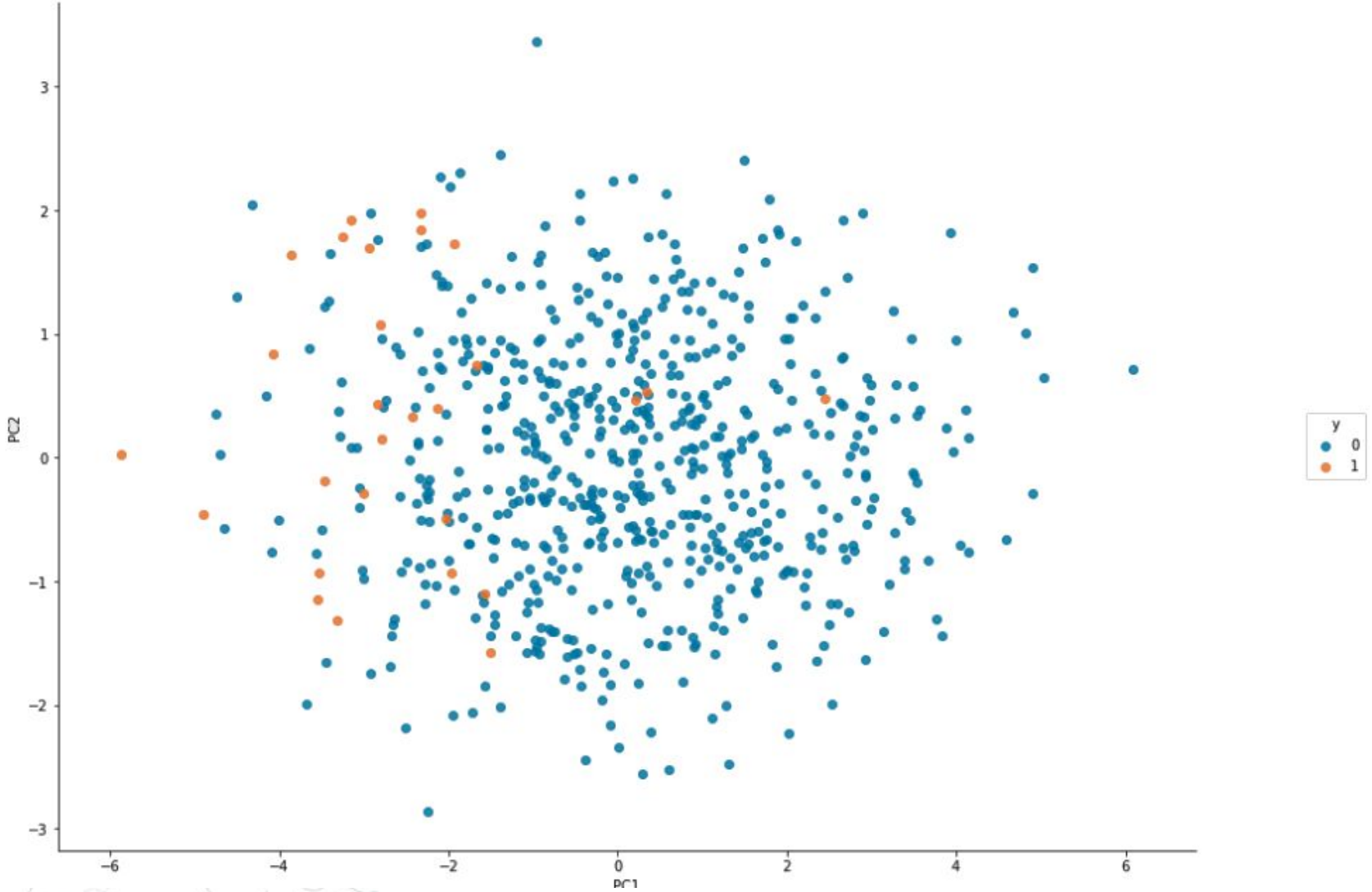
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from collections import Counter

from sklearn.datasets import make_moons, make_classification
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.model_selection import train_test_split

from imblearn import under_sampling
from imblearn import under_sampling
from imblearn import combine
```

2.
The accuracy paradox

Lets generate some data..

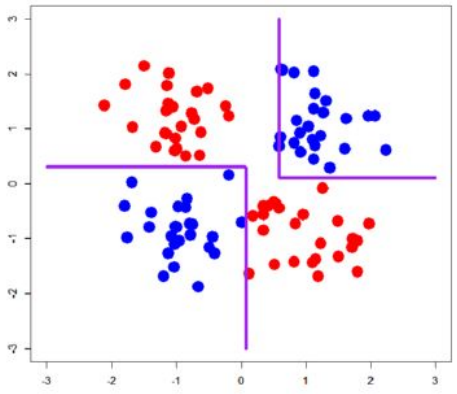
```
# Generate the dataset
X, y = make_classification(class_sep=1.3, n_samples=5000, n_features=7,
                           n_informative=3, n_redundant=1, weights=[0.95, 0.05], n_clusters_per_class=1,
                           random_state=1)

# PCA
pca = PCA(n_components=2, random_state=42)
X_ = pca.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_, y, test_size=0.33)

# Transform to pandas dataframe
df = pd.DataFrame(X_test, columns=['PC1', 'PC2'])
df['y'] = y_test
```

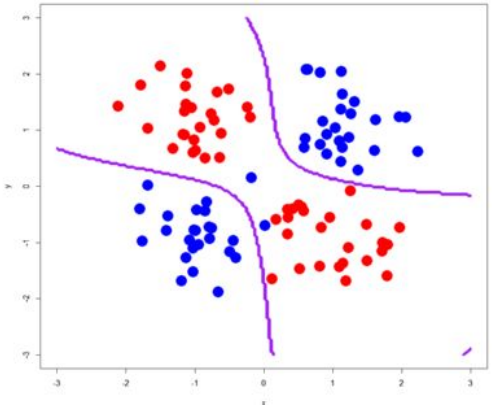
```
# Plot the data
sns.lmplot('PC1', 'PC2', df, hue='y', fit_reg=False)
fig = plt.gcf()
fig.set_size_inches(15,10)
plt.show()
```

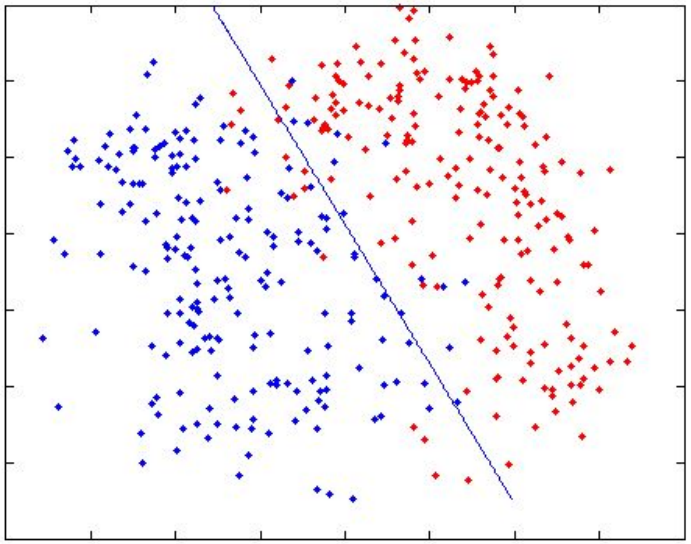


Decision boundary

Decision boundary or **decision surface** is a hypersurface that partitions the underlying vector space into two sets, one for each class.



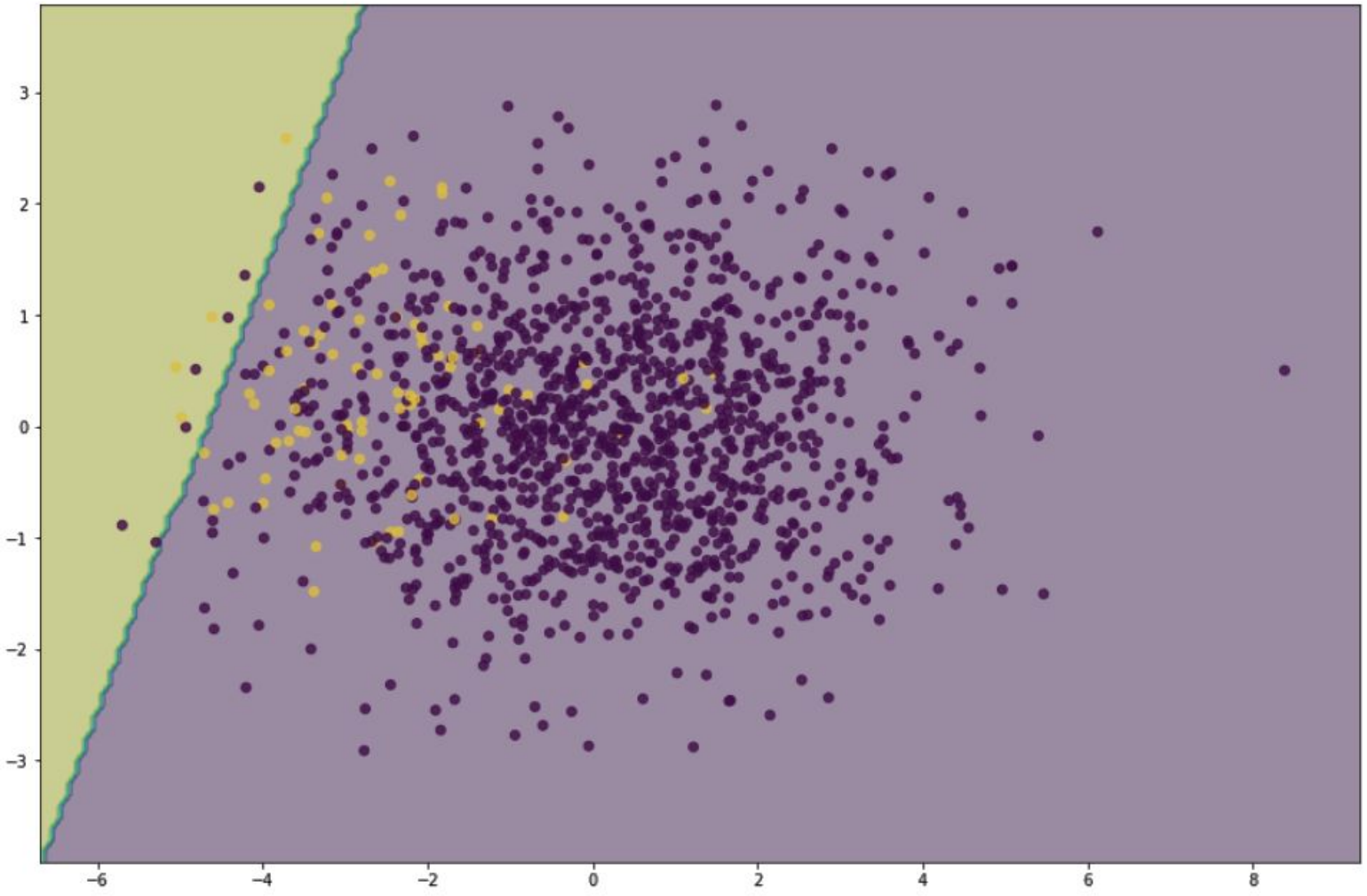




```python
def fitAndPlot(sampler, model, X_train, y_train, X_test, y_test):
    X_sampled = X_train
    y_sampled = y_train

    if sampler:
        X_sampled, y_sampled = sampler.fit_sample(X_train, y_train)

        print(f'Original dataset shape {Counter(y_train)}')
        print(f'Sampled dataset shape {Counter(y_sampled)}')

    # Fit a linear model
    model.fit(X_sampled, y_sampled)

    plot_decision_boundaries(X_sampled, y_sampled, model)

    print(metrics.confusion_matrix(y_test, model.predict(X_test), labels=[1,0]))
    print(f'\nAUC score: {metrics.roc_auc_score(y_test, model.predict(X_test))}')
    print(f'Geometric mean score: {geometric_mean_score(y_test, model.predict(X_test))}')
```

What percent of our predictions were correct?

```
sampler = None
model = LogisticRegression()

fitAndPlot(sampler=sampler, model=model, X_train=X_train, y_train=y_train, X_test=X_test, y_test=y_test)
```



Accuracy: 94%

We can do even better..

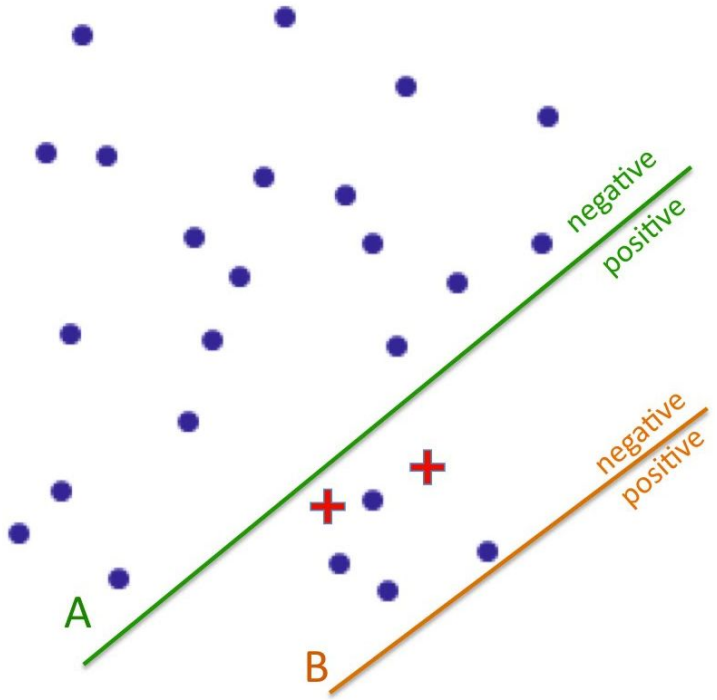
```
def my_classifier_is_better_than_yours():
    return False
```

Accuracy: 95%

The accuracy paradox

We are predicting if a user is going to click on an ad banner

- Which is a better classifier
  A or B?
- Which has a higher accuracy rate?
- **Accuracy is a bad metric in cases like this**



3. Alternative performance metrics

The confusion matrix

Our model:

```
[[  1  27]
 [  4 628]]
```

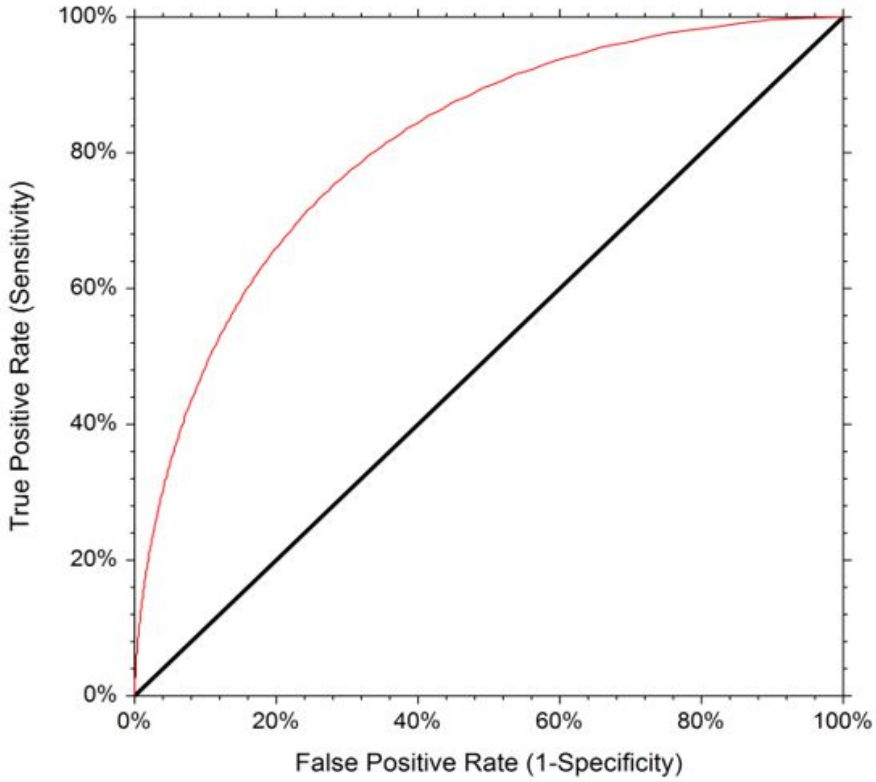
$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}$$

$$\text{sensitivity} = \frac{\text{number of true positives}}{\text{number of true positives} + \text{number of false negatives}}$$

$$\text{specificity} = \frac{\text{number of true negatives}}{\text{number of true negatives} + \text{number of false positives}}$$

	p' (Predicted)	n' (Predicted)

ROC curve



AUC (Area under the curve)

the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one

What is our naive models AUC score?

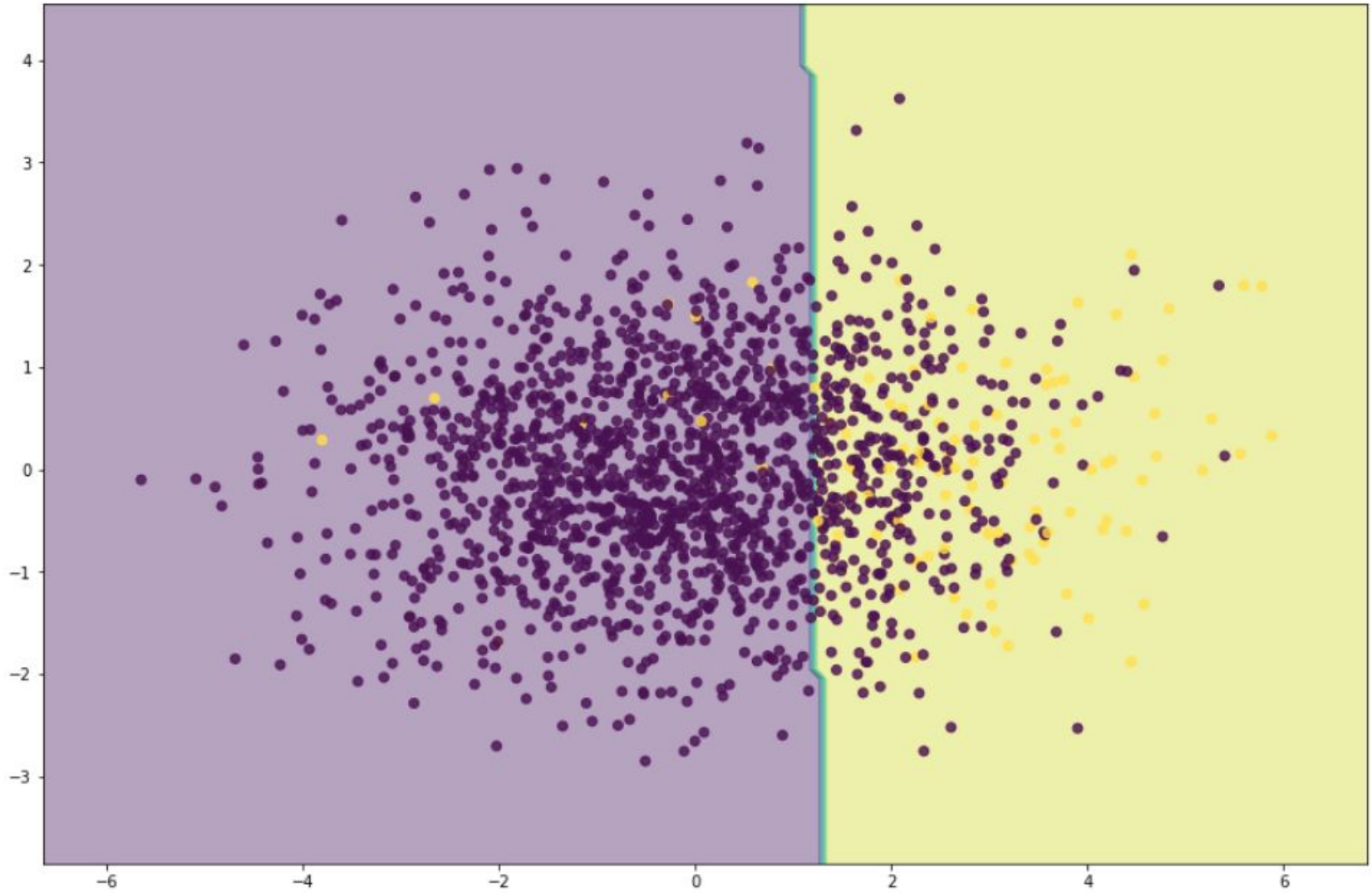
AUC score: 0.562

$$\text{AUC} = \begin{cases} 0.5 & \text{no discrimination} \\ 0.6\text{–}0.7 & \text{poor} \\ 0.7\text{–}0.8 & \text{accceptable (fair)} \\ 0.8\text{–}0.9 & \text{excellent (good)} \\ > 0.9 & \text{outstanding} \end{cases}$$

Geometric mean score: 0.356

3.
Ratio parameter

```
# Fit a linear model
lmClassWeight = LogisticRegression(class_weight={0: 0.05, 1:0.95})
lmClassWeight.fit(X_train, y_train)

plot_decision_boundaries(X_test, y_test, lmClassWeight)
```

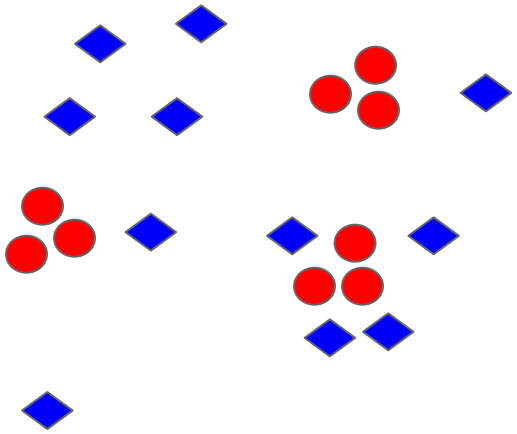


AUC score: 0.814
Geometric mean score: 0.812

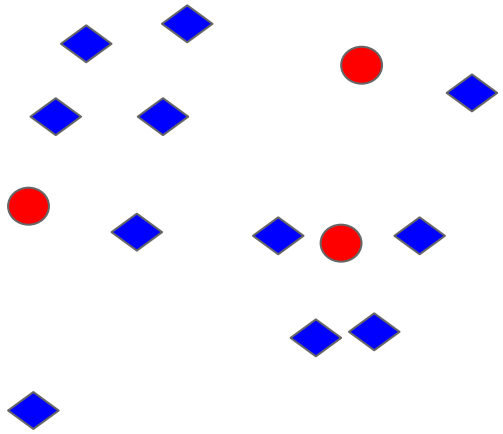
4.
Resampling

Resampling

Over



Under



Resampling

Over

AUC score:
0.819
Geometric mean score:
0.810

Under

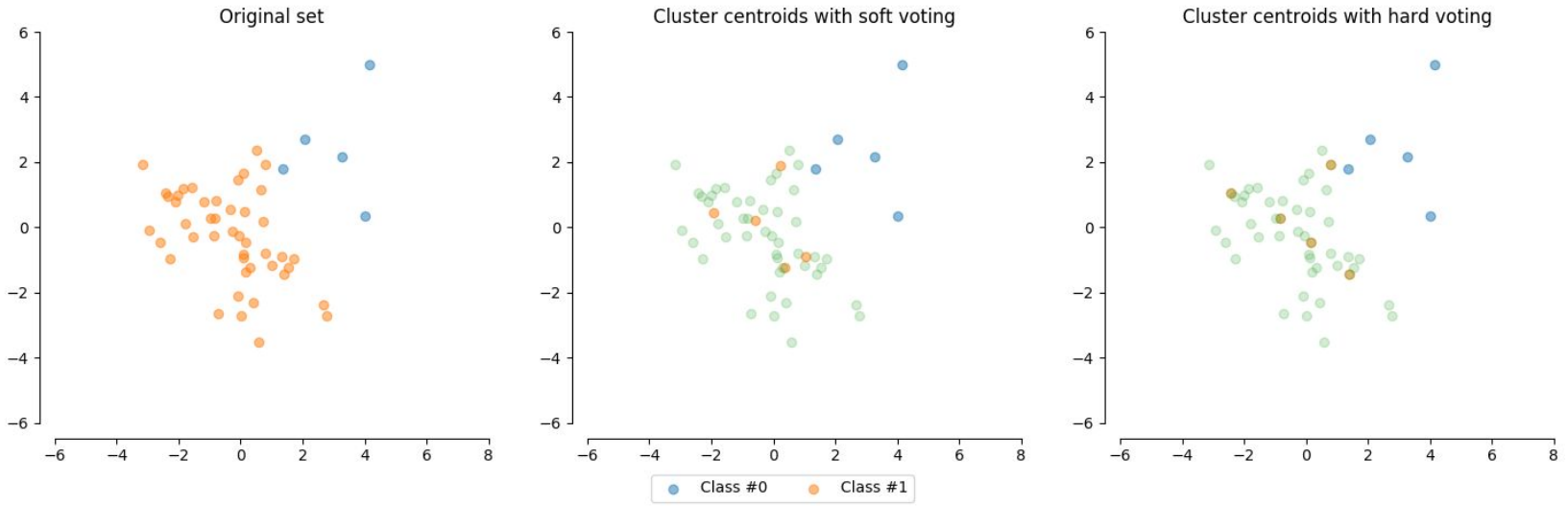
AUC score:
0.816
Geometric mean score:
0.815

Cluster Centroids

Replacing a cluster of majority samples by the cluster centroid of a K-Means algorithm

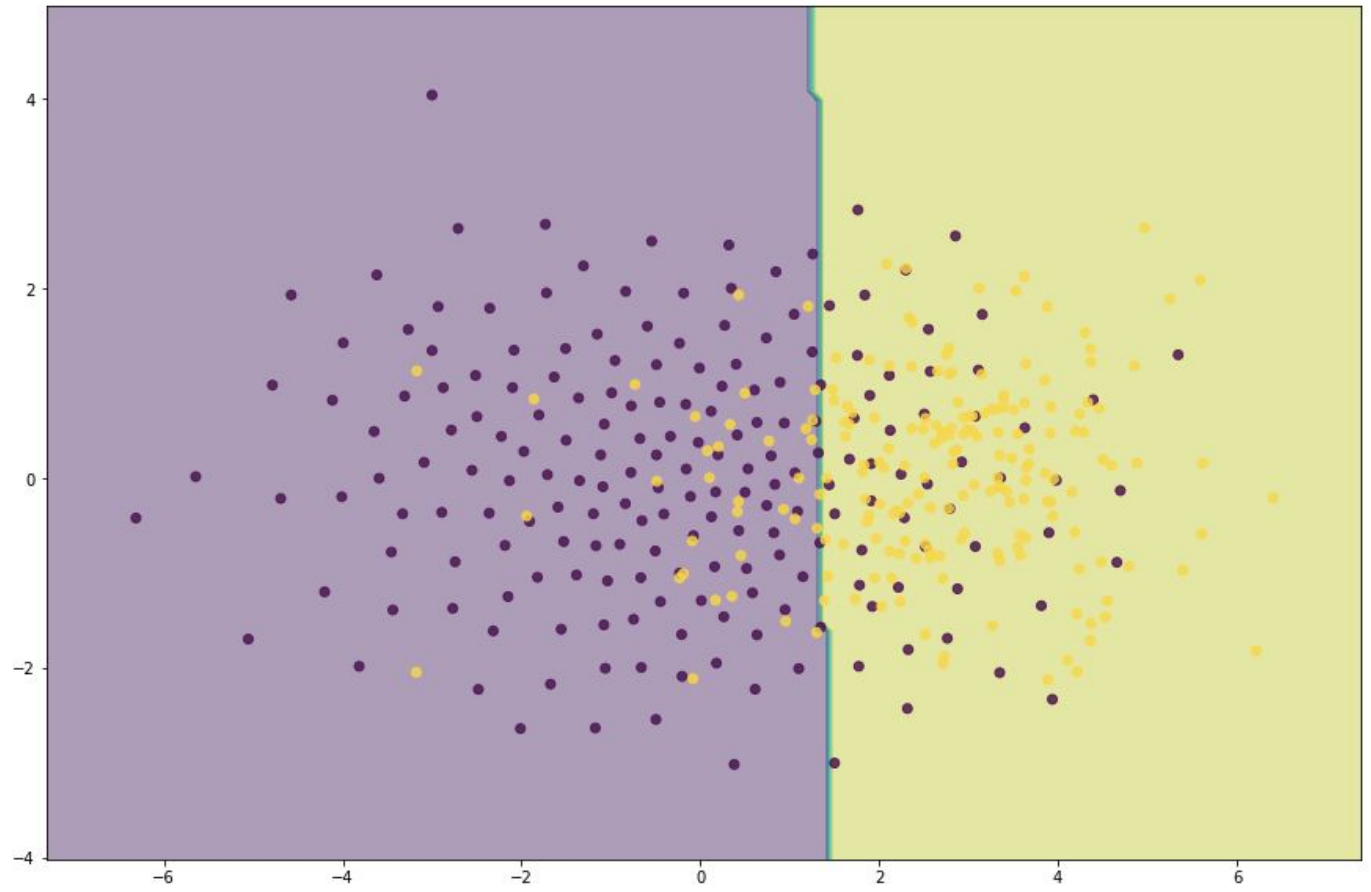
Advantage:

◎ Avoid the information loss of majority class.



```
from imblearn.under_sampling import ClusterCentroids

sampler = ClusterCentroids()
model = LogisticRegression()

fitAndPlot(sampler=sampler, model=model, X_train=X_train, y_train=y_train, X_test=X_test, y_test=y_test)
```

```
Original dataset shape Counter({0: 3150, 1: 200})
Sampled dataset shape Counter({0: 200, 1: 200})
```




```
[[  53   22]
 [ 290 1285]]
```

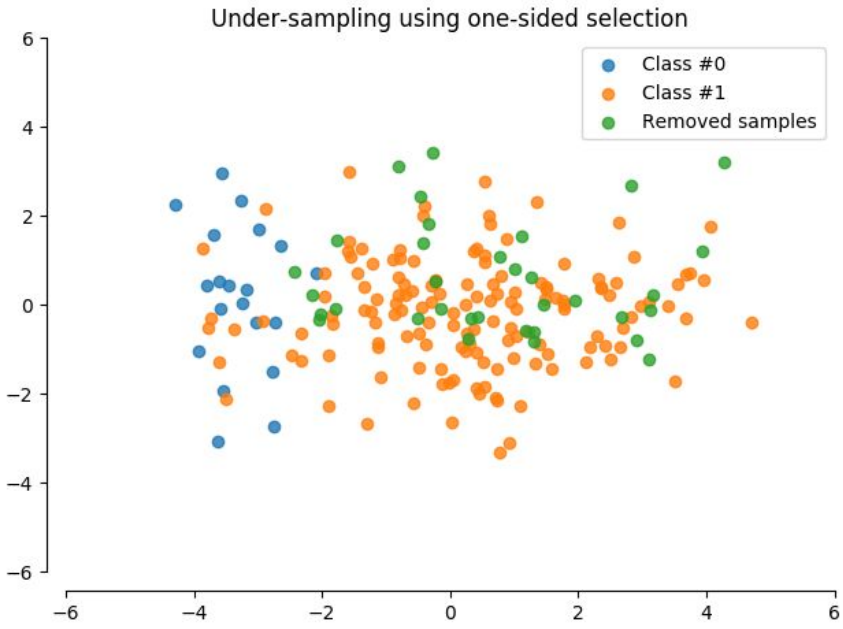
AUC score: 0.806

Geometric mean score: 0.8

One-Sided Selection

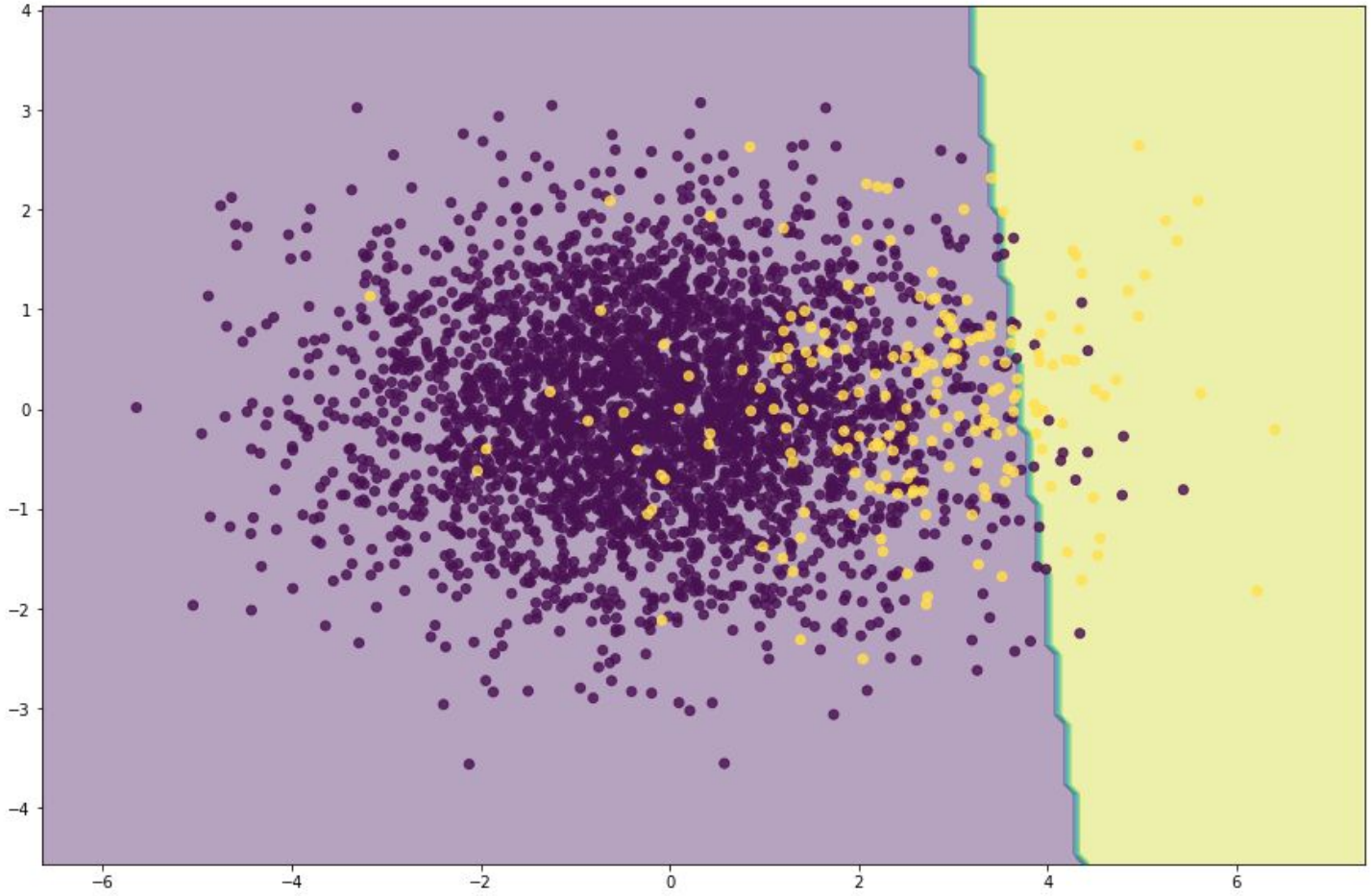
The majority samples are roughly divided into 4 groups:

- ***Noise examples***: close to any minority example.
- ***Borderline examples***: close to the boundary. unreliable.
- ***Redundant examples***: can not provide any other useful information.
- ***Safe examples***.



```
from imblearn.under_sampling import OneSidedSelection
sampler = OneSidedSelection()
model = LogisticRegression()

fitAndPlot(sampler=sampler, model=model, X_train=X_train, y_train=y_train, X_test=X_test, y_test=y_test)
```

```
Original dataset shape Counter({0: 3158, 1: 192})
Sampled dataset shape Counter({0: 3064, 1: 192})
```



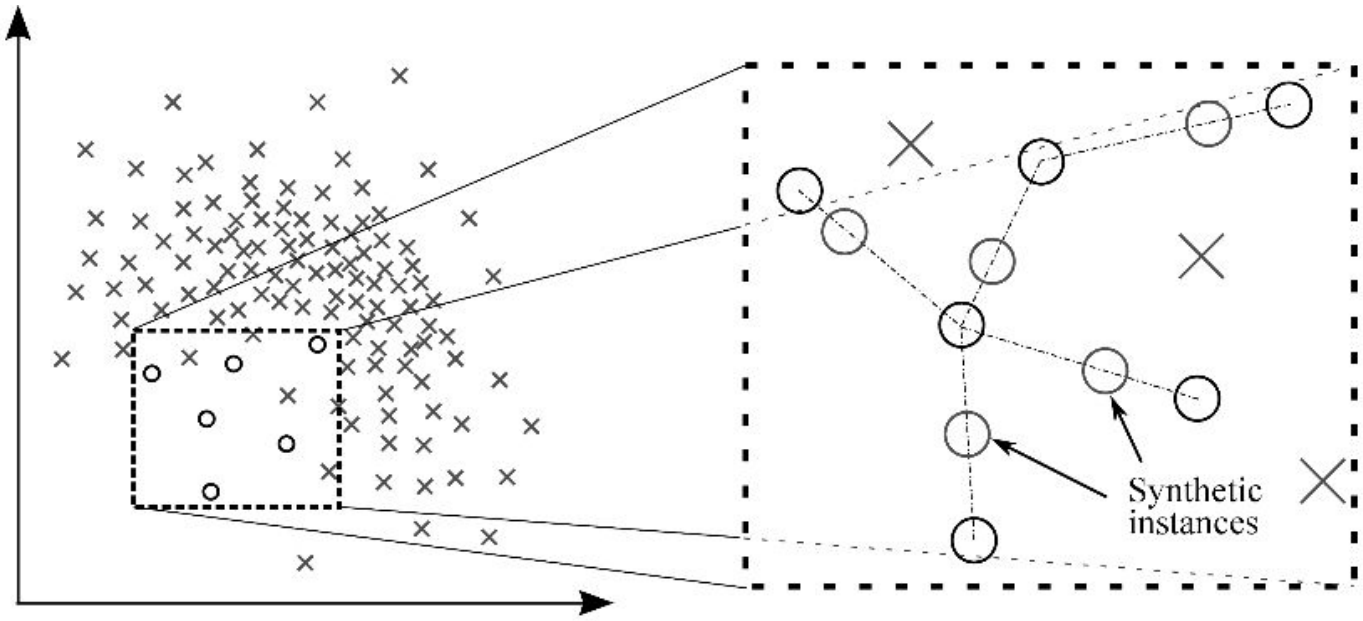

```
[[  22   61]
 [  11 1556]]
```

AUC score: 0.613
Geometric mean score: 0.482

SMOTE: Synthetic Minority Oversampling Algorithm

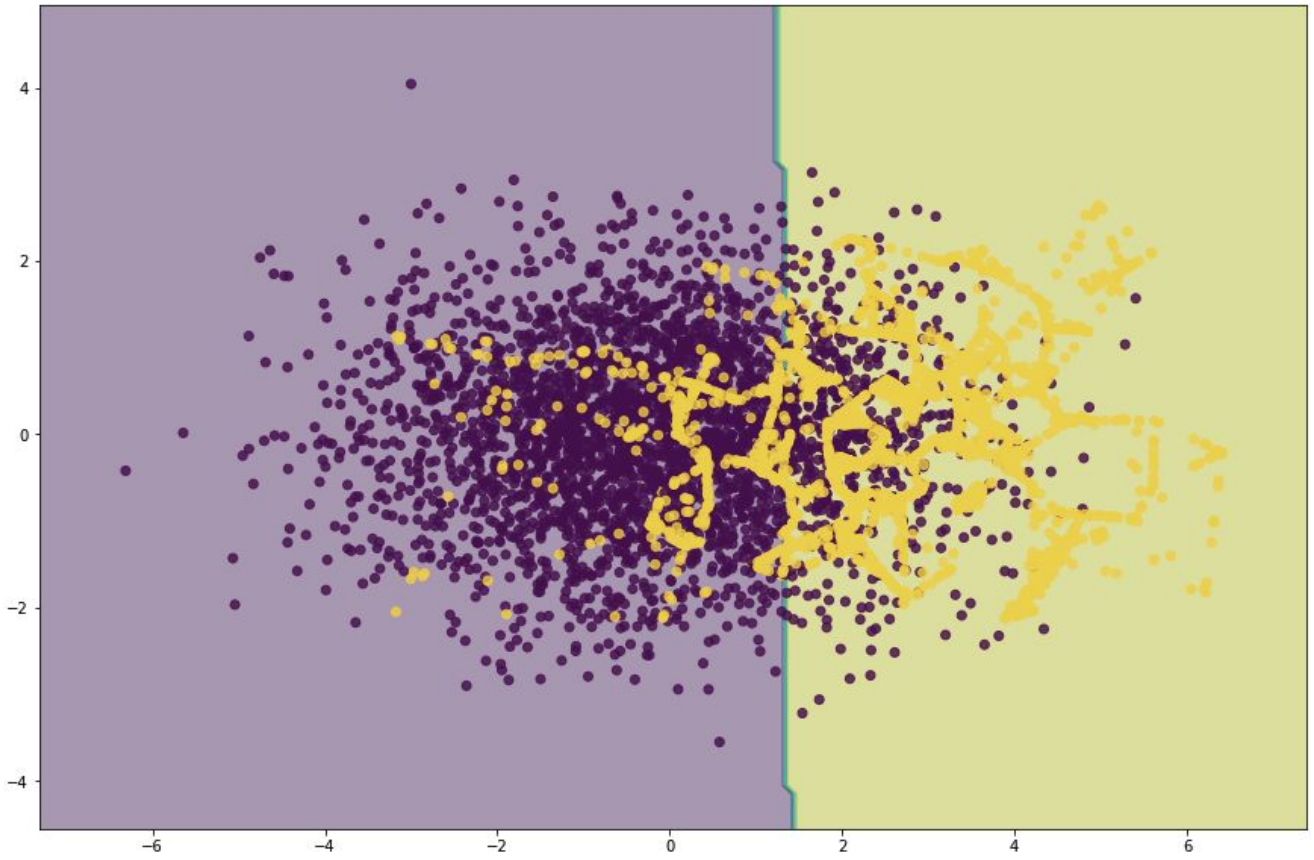
Generates new minority class samples along the lines between S and each nearest minority neighbor.

- Avoid overfitting which occurs when exact replicas of minority instances are added in random oversampling



```
from imblearn.over_sampling import SMOTE
sampler = SMOTE()
model = LogisticRegression()

fitAndPlot(sampler=sampler, model=model, X_train=X_train, y_train=y_train, X_test=X_test, y_test=y_test)
```

```
Original dataset shape Counter({0: 3150, 1: 200})
Sampled dataset shape Counter({1: 3150, 0: 3150})
```




```
[[  54   21]
 [ 299 1276]]
```

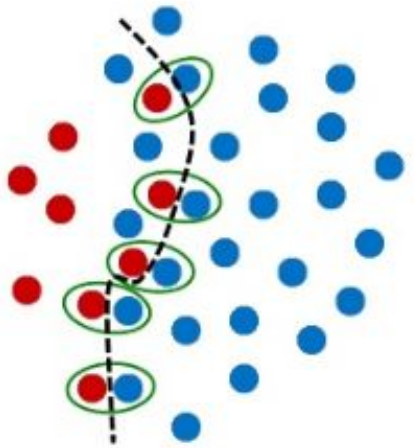
AUC score: 0.814

Geometric mean score: 0.814

Tomec links

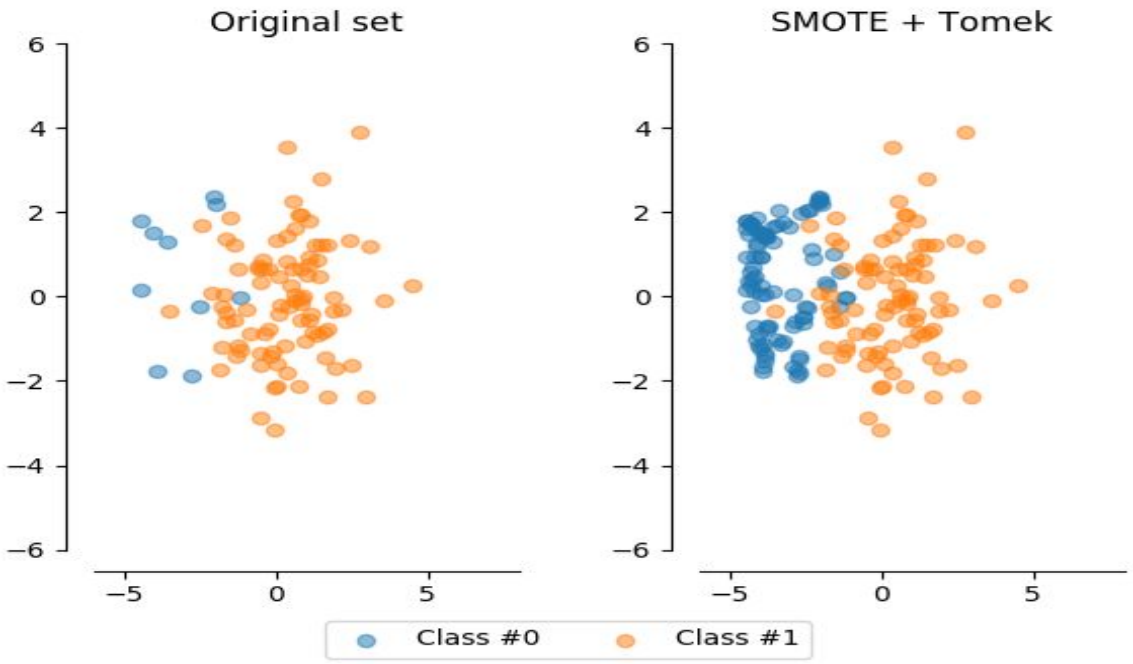
- Can be used as an under-sampling method or as a data cleaning method.

- When used with SMOTE it is used on the over-sampled training set as a data cleaning method.



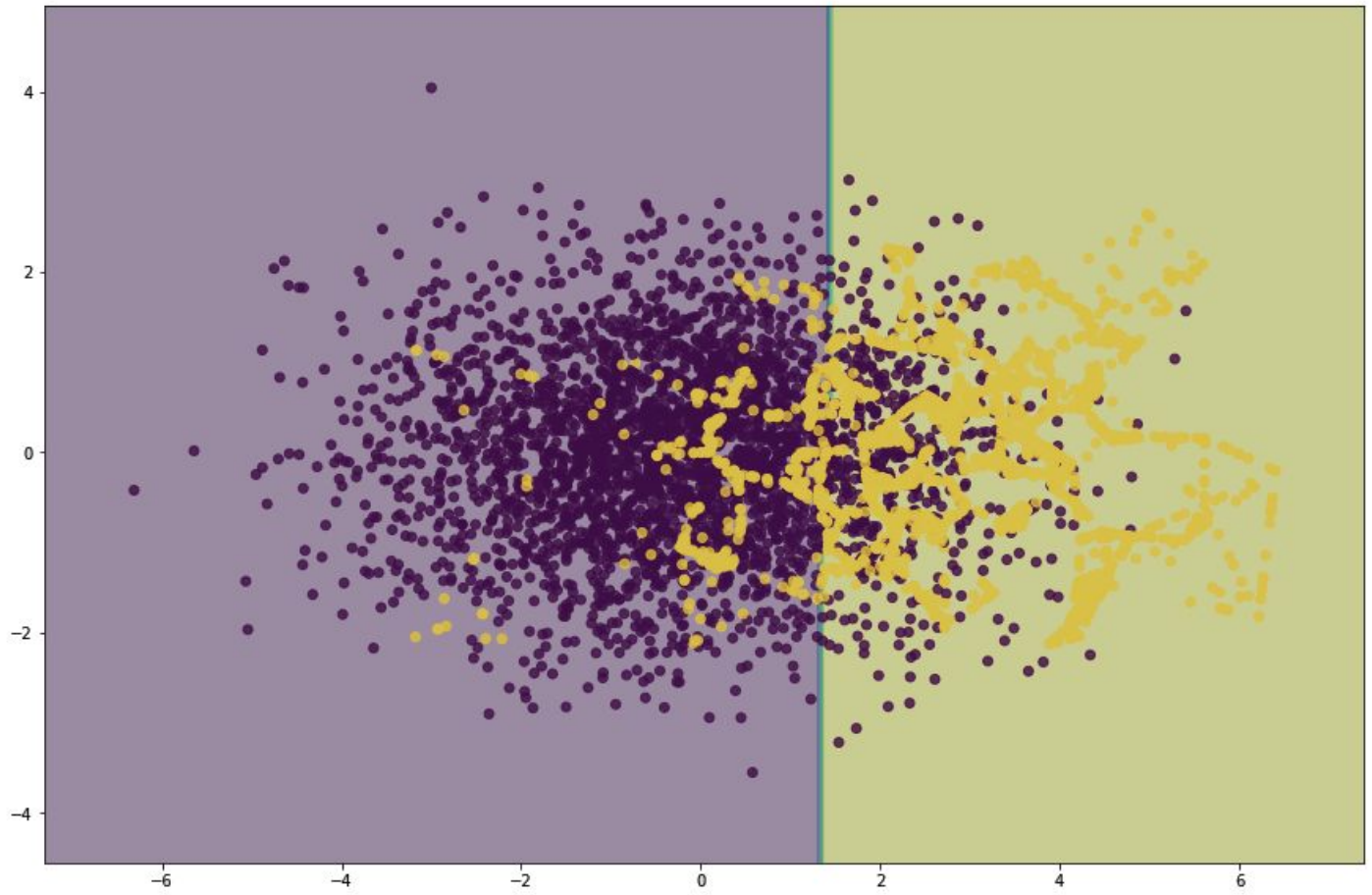
SMOTE + Tomec

- Avoid overfitting
- New synthetic similar instances are created
- The new dataset is used as a sample to train the classification models.



```
from imblearn.combine import SMOTETomek
sampler = SMOTETomek()
model = LogisticRegression()

fitAndPlot(sampler=sampler, model=model, X_train=X_train, y_train=y_train, X_test=X_test, y_test=y_test)
```

```
Original dataset shape Counter({0: 3150, 1: 200})
Sampled dataset shape Counter({1: 2948, 0: 2948})
```




```
[[  54   21]
 [ 284 1291]]
```

AUC score: 0.82
Geometric mean score: 0.82

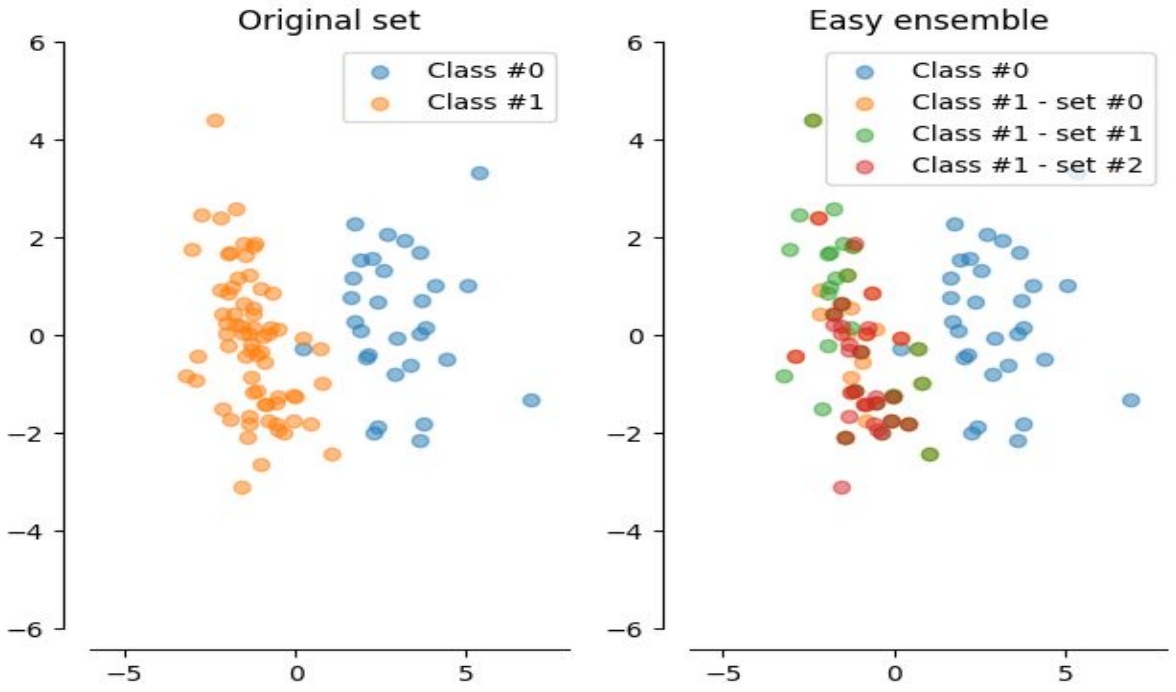
5.

Ensemble methods

Easy Ensemble

Create an ensemble sets by iteratively applying random under-sampling.

- select a random subset and make an ensemble of the different sets.



Some more techniques

Resampling:

- **Border SMOTE**: generates the synthetic sample along the borderline of minority and majority classes.
- **DBSMOTE**: Density-Based Synthetic Minority Over-sampling Technique is based on clustering algorithm DBSCAN.
- **ANS:** dynamically adapts the number of neighbors needed for oversampling around different minority regions. **parameter free.**

Ensemble

- **BalanceCascade**: ensure that misclassified samples can again be selected for the next subset. **the classifier play the role of a "smart" replacement method.**

Summary

- Ratio parameter
- Over/Under Sampling
- Ensemble

Thanks!

Any questions?

zehori.ido@gmail.com
idozehori.com