imbalanced-learn sampler

In [ ]:

```
%matplotlib inline
```

In [ ]:

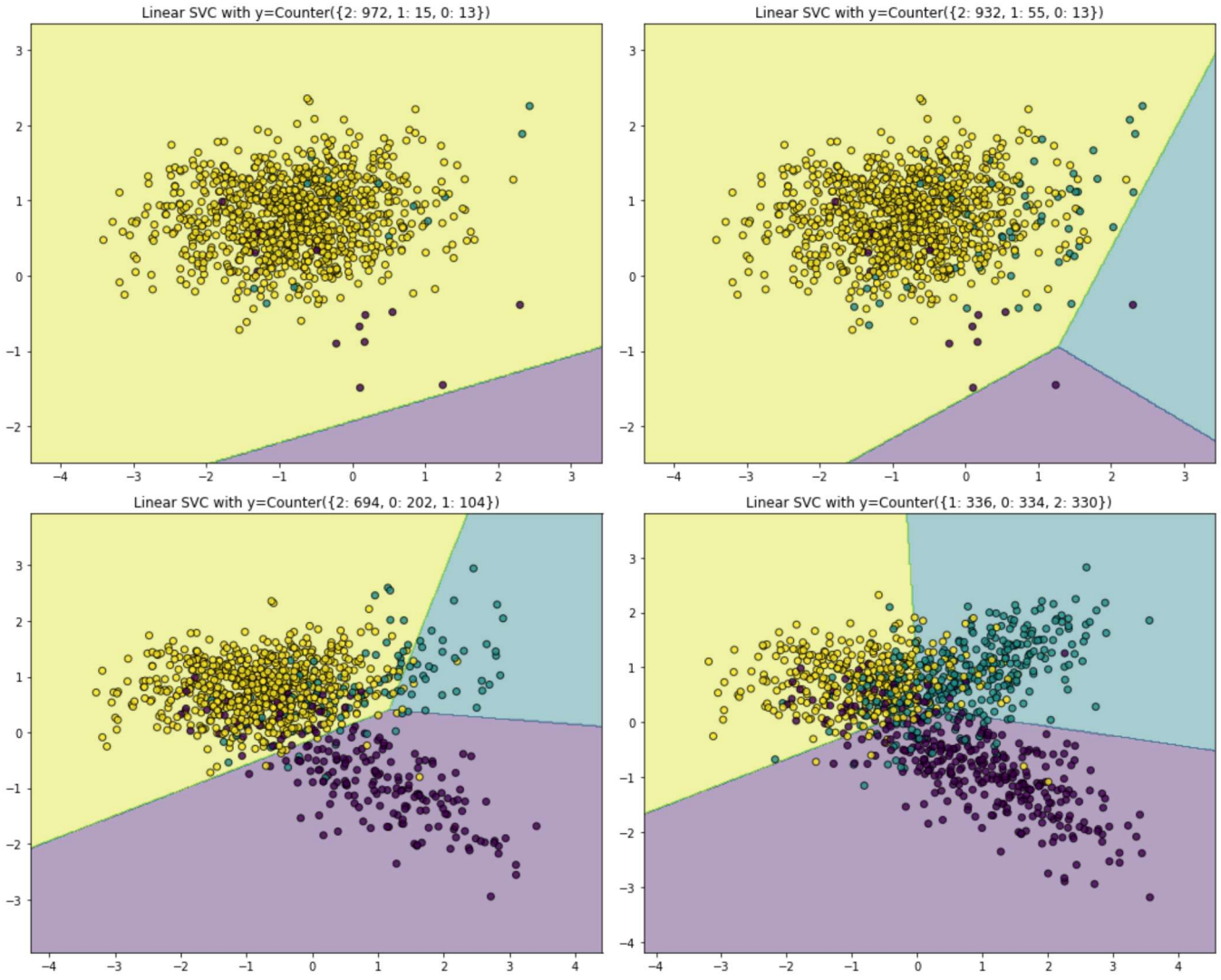
```
%run sample1.py
```

Balancing issue

- difference of the number of samples in the different classes
- e.g. effect of training a linear SVM classifier with different level of class balancing
  - decision function of the linear SVM is highly impacted
  - with a greater imbalanced ratio, the decision function favor the majority class

In [ ]:

```
sample1()
```

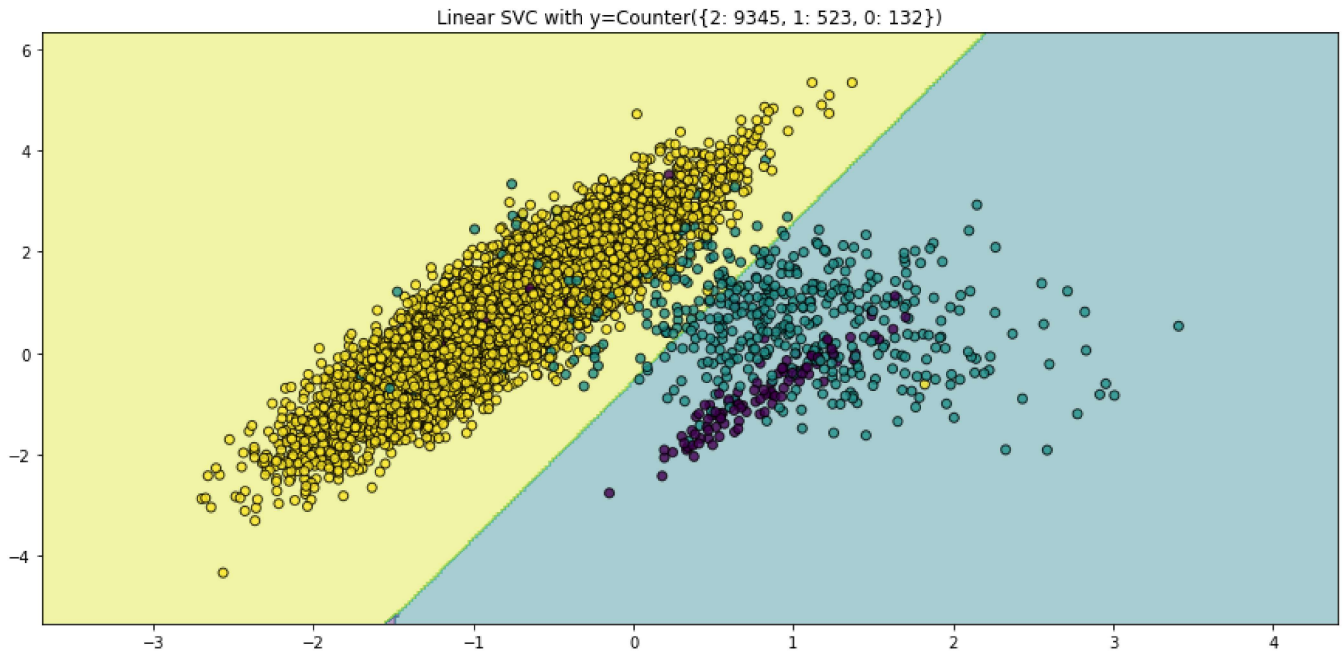


Over-sampling

- generate new samples in the classes which are under-represented
  - Random Sampling
  - SMOTE(Synthetic Minority Oversampling Technique)
  - ADASYN(Adaptive Synthetic)

In [ ]:

```
original()
```

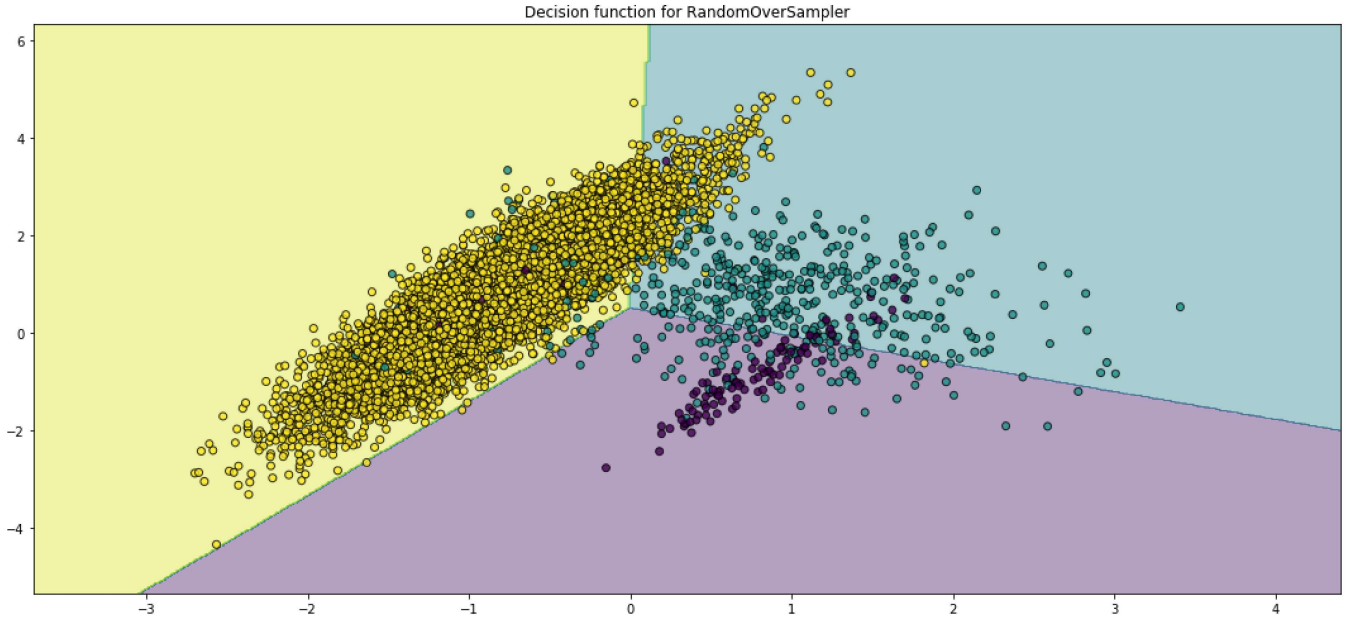


RandomOverSampler

- generate new samples by randomly sampling with replacement the current availbale samples
- the augmented dataset should be used instead of original dataset to train a classfier

In [ ]:

```
ramdomsample()
```



SMOTE

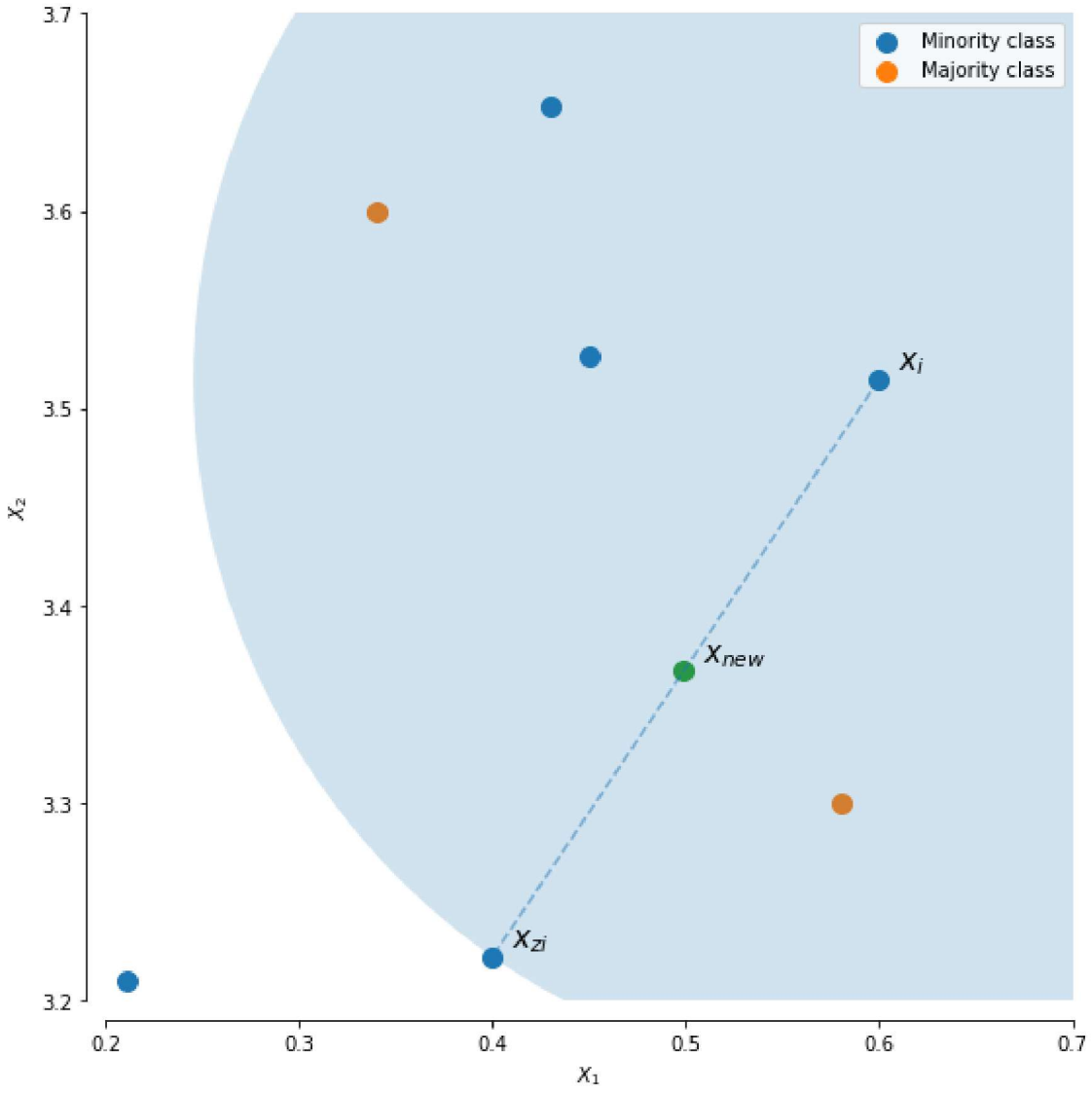
- from sample x_i, a new sample x_{new} will be generated considering its k neareast-neighbors
- λ is a random number in the range [0, 1]

$$x_{new} = x_i + \lambda \times (x_{zi} - x_i)$$

- regular SMOTE: randomly pick-up all possible x_i

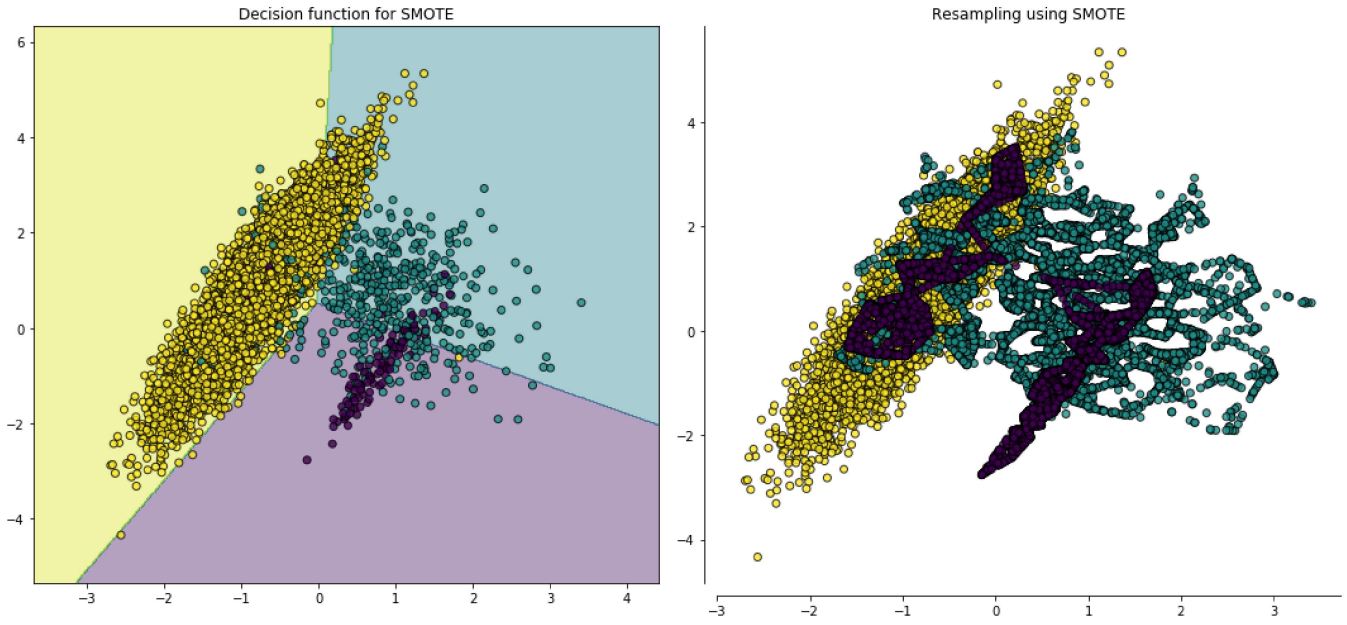
In [ ]:

```
oversample_algo()
```


Minority class
Majority class
x_i
x_{new}
x_{zi}
x_1
x_2
3.7
3.6
3.5
3.4
3.3
3.2
0.2
0.3
0.4
0.5
0.6
0.7

In [ ]:

```
df_smote()
```


Decision function for SMOTE
Resampling using SMOTE

ADASYN

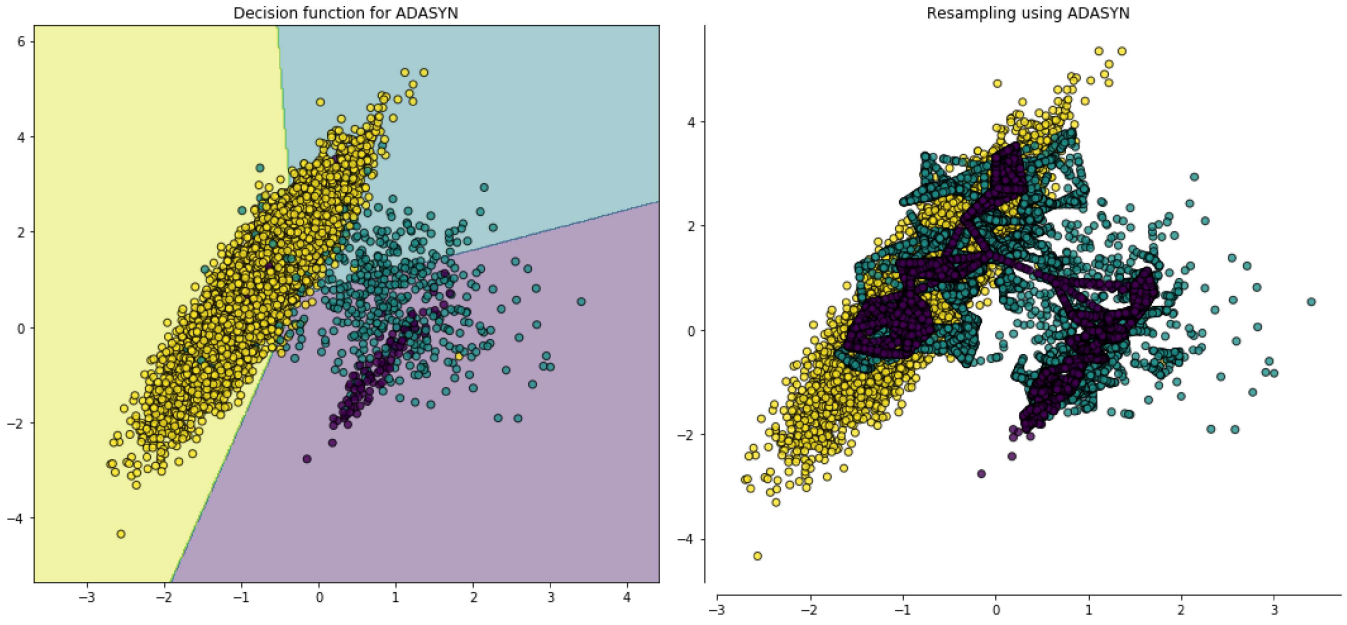
- from sample x_i, a new sample x_{new} will be generated considering its k neareast-neighbors
- λ is a random number in the range [0, 1]

$$x_{new} = x_i + \lambda \times (x_{zi} - x_i)$$

- number of samples generated form each x_i is proportiaonal to the number of samples which are not from the same class than x_i in a given neighborhood
- focus on the samples which are difficult to classify with a nearest-neighbors rule

In [ ]:

```
df_adasyn()
```

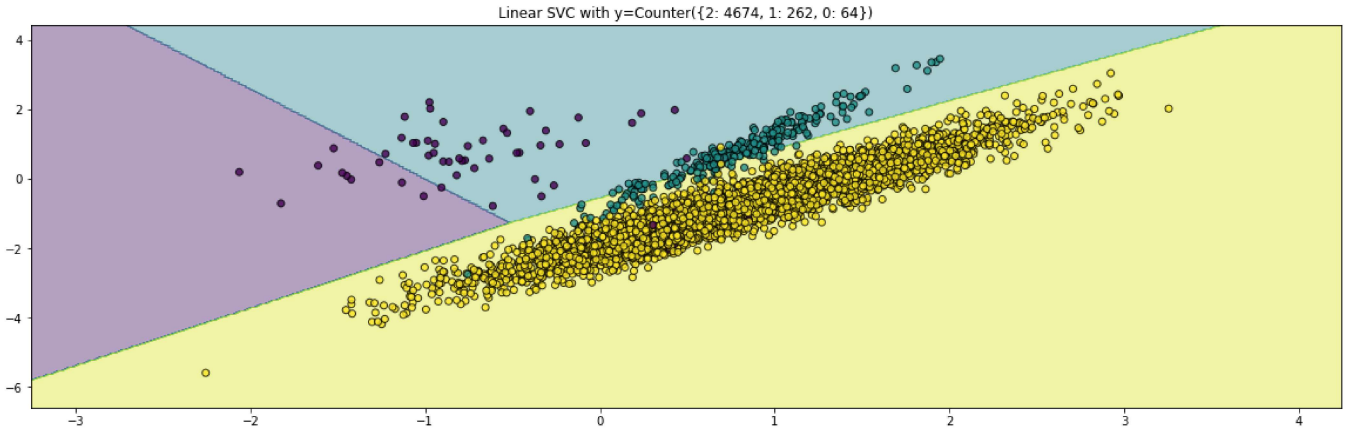


Unser-sampling

- Prototype generation:under-sampling by generating new samples
- Prototype selection: under-sampling by selecting existing samples

In [ ]:

```
gen_original()
```

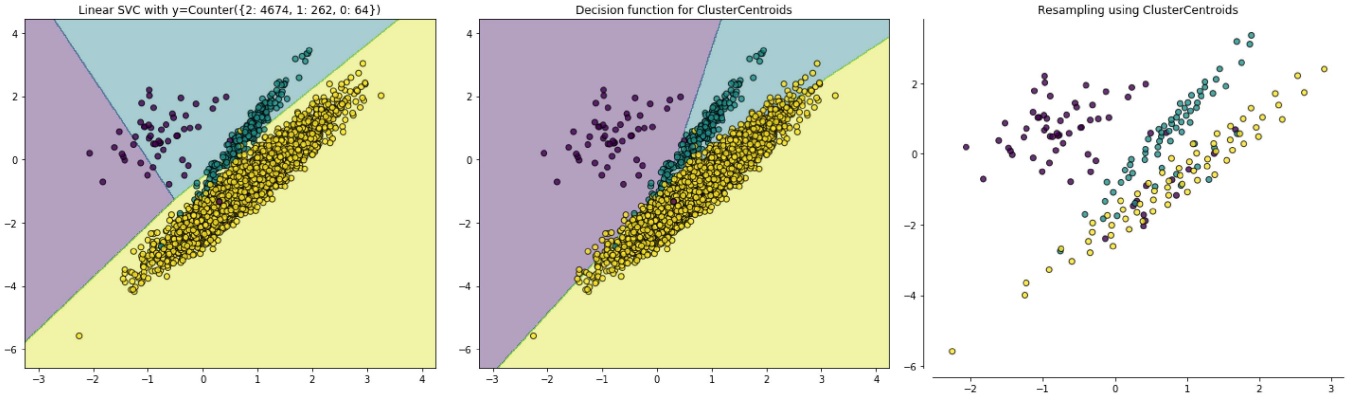


Prototype generation

- generate a new set S' where |S'| < |S| and S' $\notin$ S
- `ClusterCentroids`

In [ ]:

```
gen_undersample()
```

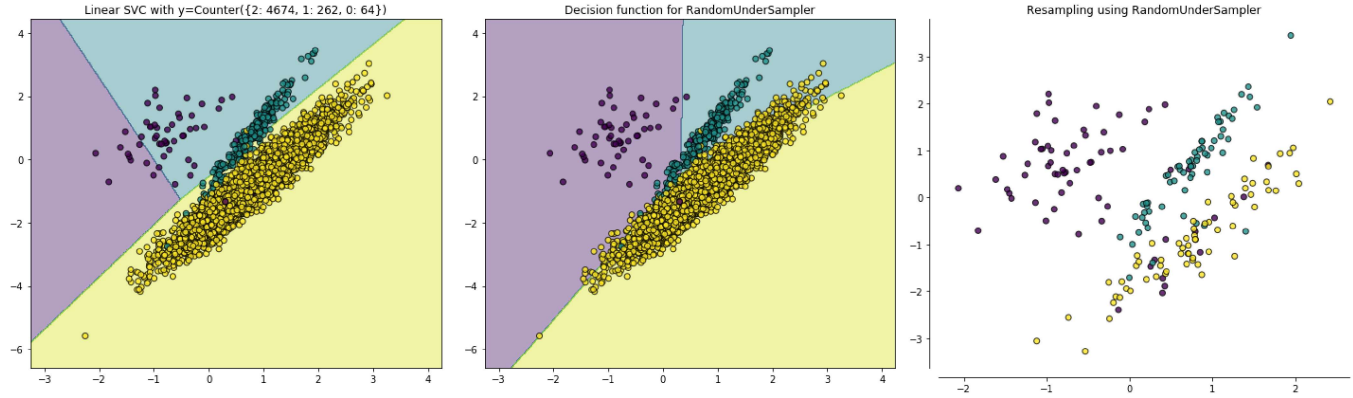


Prototype selection

- select samples from the original set S. Therefore, S' is defined such as |S'| < |S| and S' $\in$ S.

In [ ]:

```
sel_undersample()
```

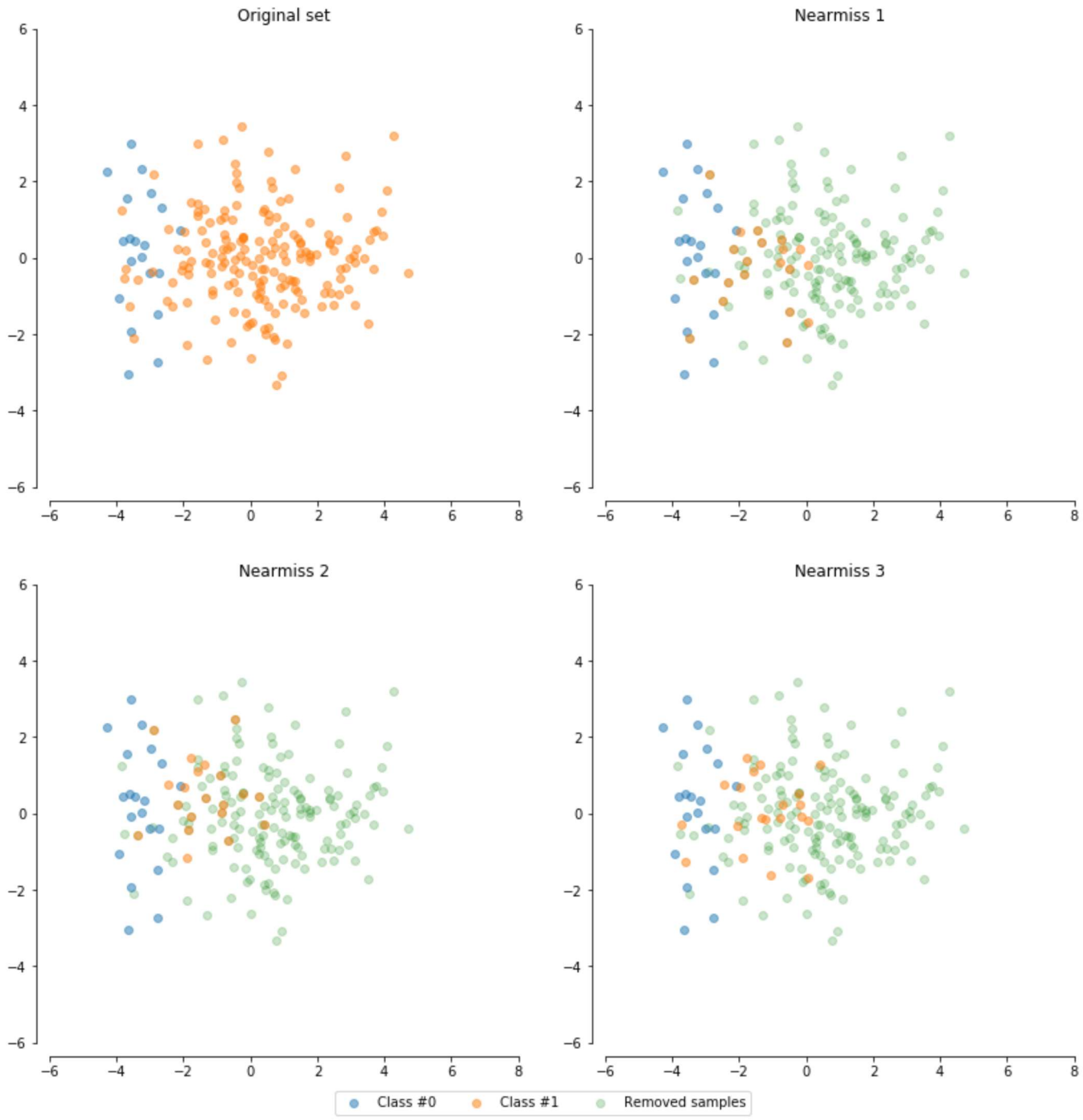


Techniques

- controlled under-sampling techniques
  - number of samples in S' is specified by the user
  - NearMiss
    - adds some heuristic rules(knn) to select samples
    - version = 1, 2, 3 (size of nn to consider to compute the average distance to the minority point samples)

In [ ]:

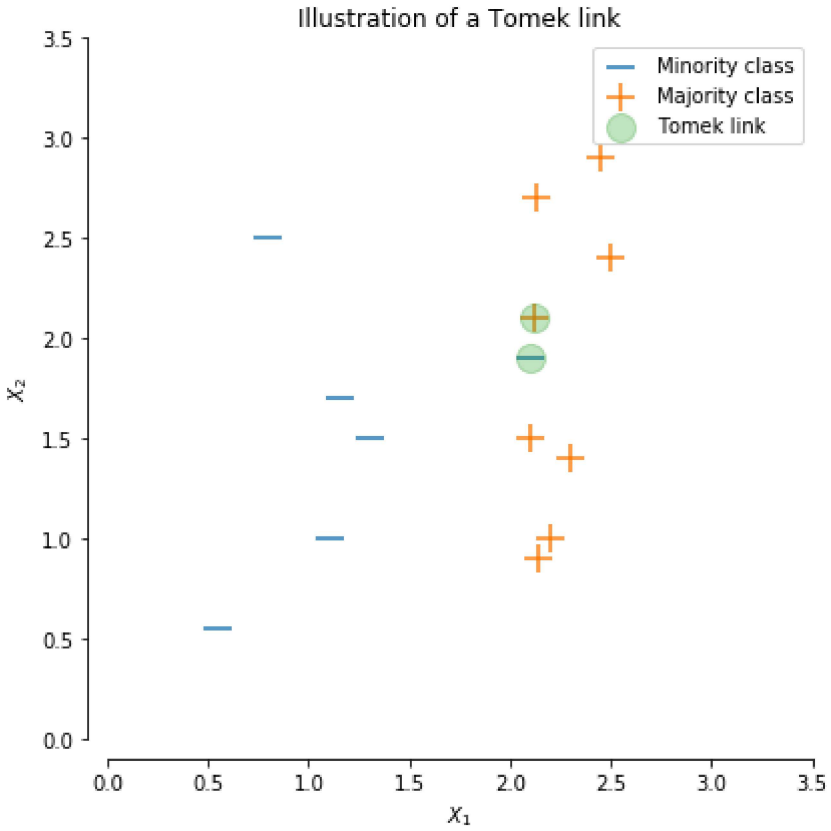
```
ex_nearmiss()
```



- cleaning under-sampling techniues
  - don't allow specify the number of samples to have in each class
  - TomekLnks
    - exist if the two samples are the nearest neighbors of each other

In [ ]:

```
ex_tomek()
```

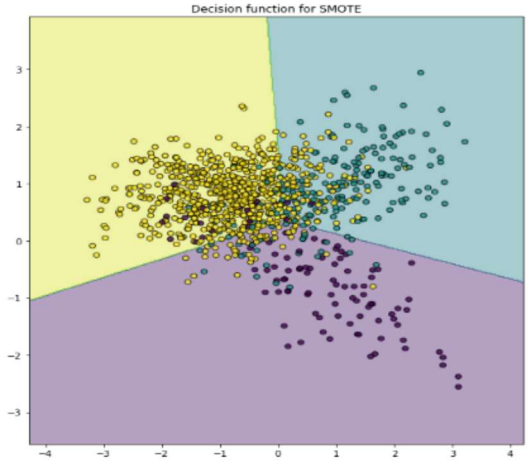


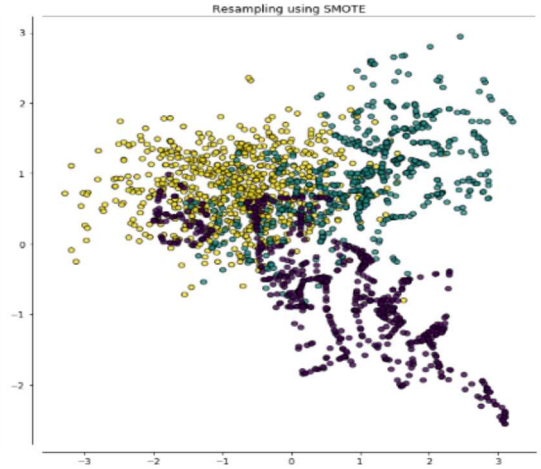
Combination of over-and under-sampling

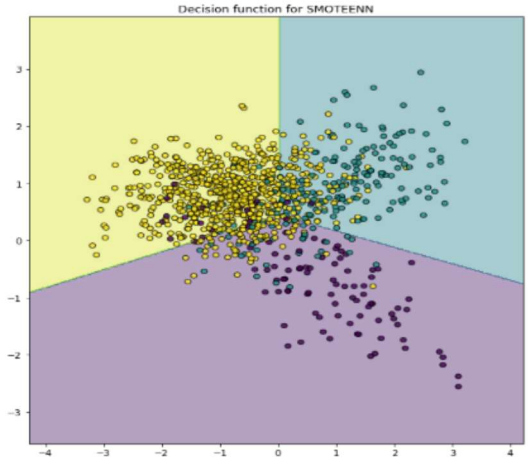
- **generate** noisy samples by interpolating new points between marginal outliers and inliers
- **cleaning** the resulted space obtained after over-sampling
  - `SMOTETomek` : SMOTE -> Tomek
  - `SMOTEENN` : SMOTE -> edited nearest-neighbours

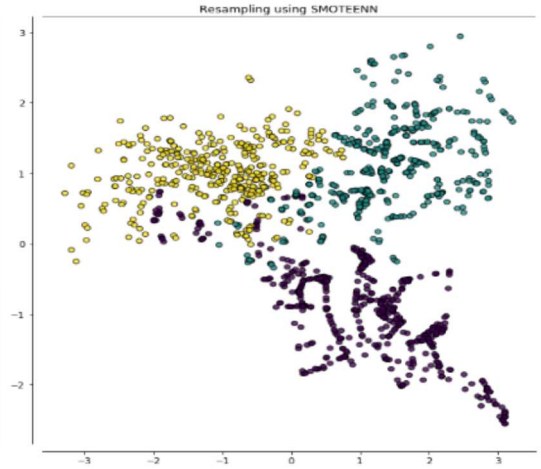
In [ ]:

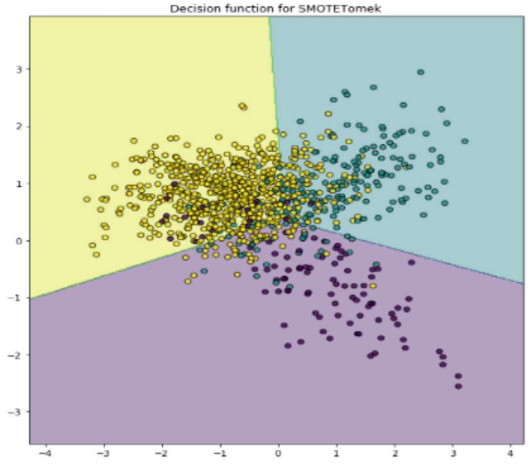
```
ex_combi()
```

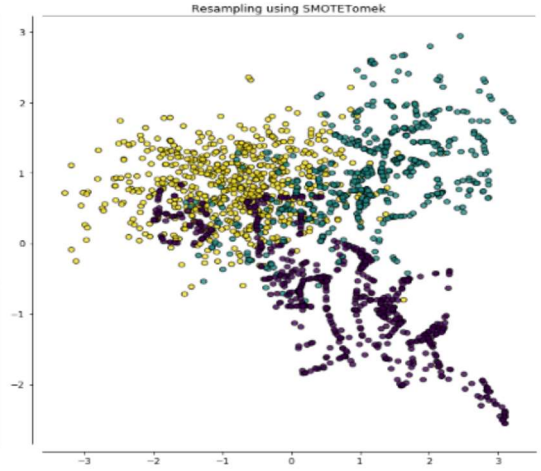

Decision function for SMOTE


Resampling using SMOTE


Decision function for SMOTEENN


Resampling using SMOTEENN


Decision function for SMOTETomek


Resampling using SMOTETomek