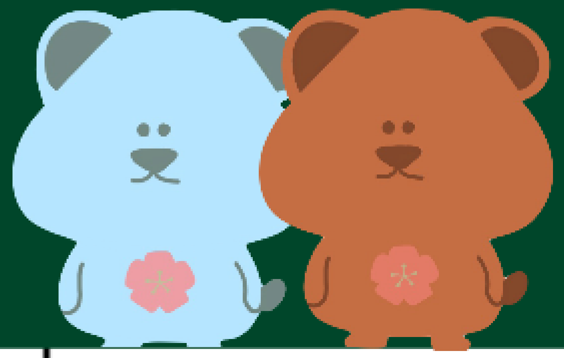
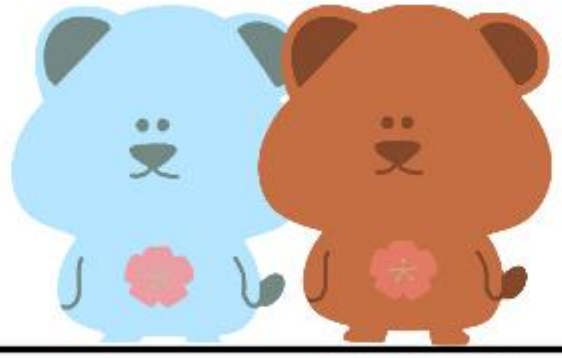


Week17: IEEE-CIS Fraud Detection

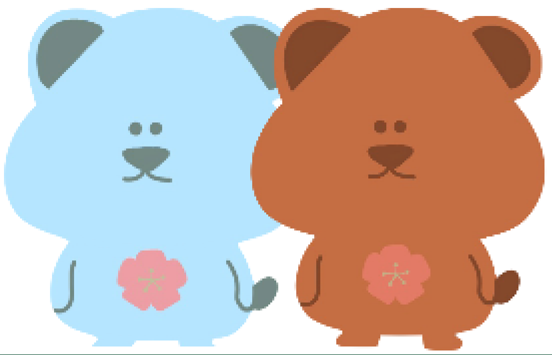
발표자: 이지호, 김희숙

목차



EWHA EURON

대회 소개



#01 대회 소개 IEEE-CIS Fraud Detection

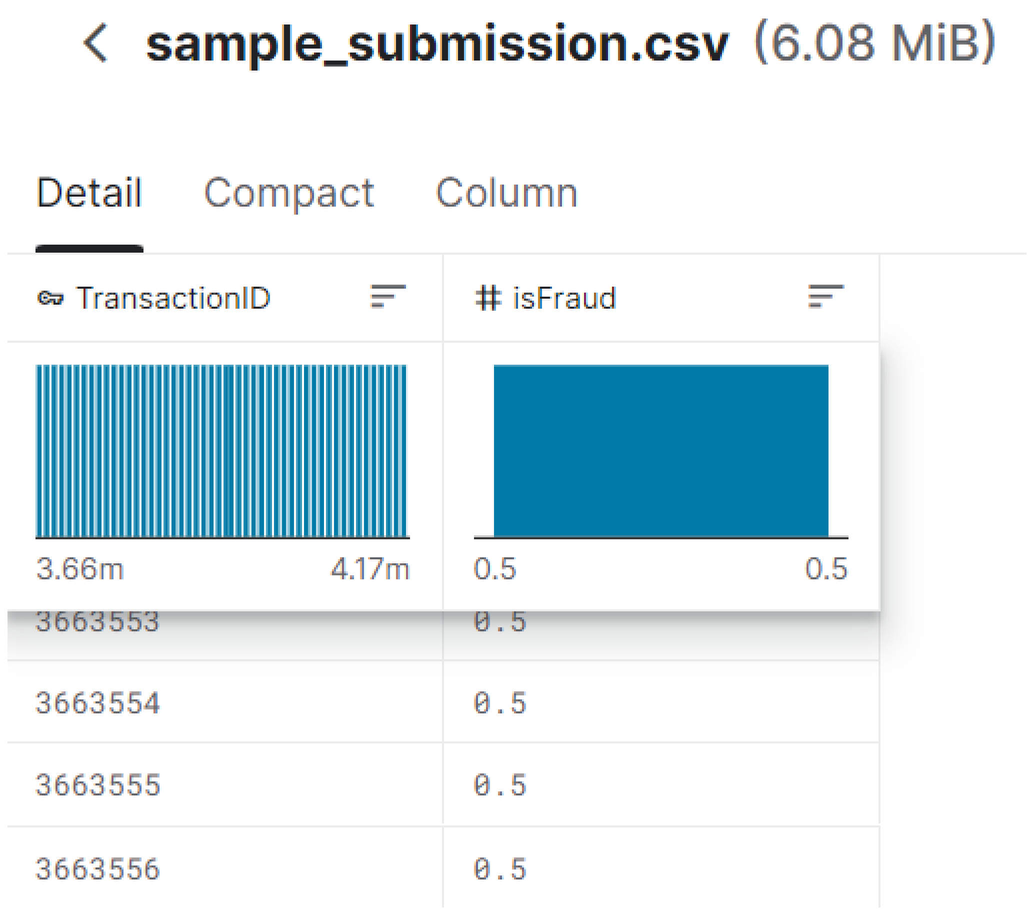
대회 개요

- Vesta 사에서 제공하는 전자 상거래 transaction 및 feature 데이터셋을 활용해 사기탐지 모델 작성하기.

데이터 셋

- Features
  - Transaction Table *

- TransactionDT: timedelta from a given reference datetime (not an actual timestamp)
- TransactionAMT: transaction payment amount in USD
- ProductCD: product code, the product for each transaction
- card1 - card6: payment card information, such as card type, card category, issue bank, country, etc.
- addr: address
- dist: distance
- P_ and (R__) emaildomain: purchaser and recipient email domain
- C1-C14: counting, such as how many addresses are found to be associated with the payment card, etc. The actual meaning is masked.
- D1-D15: timedelta, such as days between previous transaction, etc.
- M1-M9: match, such as names on card and address, etc.
- Vxxx: Vesta engineered rich features, including ranking, counting, and other entity relations.



#01 대회 소개 IEEE-CIS Fraud Detection

데이터 셋: train_transaction.csv

```
train_transactions.head()
```

	TransactionID	isFraud	TransactionDT	TransactionAmt	ProductCD
0	2987000	0	86400	68.5	W
1	2987001	0	86401	29.0	W
2	2987002	0	86469	59.0	W
3	2987003	0	86499	50.0	W
4	2987004	0	86506	50.0	H

card1	card2	card3	card4	card5	card6
13926	NaN	150.0	discover	142.0	credit
2755	404.0	150.0	mastercard	102.0	credit
4663	490.0	150.0	visa	166.0	debit
18132	567.0	150.0	mastercard	117.0	debit
4497	514.0	150.0	mastercard	102.0	credit

addr2	dist1	dist2	P_emaildomain	R_emaildomain
87.0	19.0	NaN	NaN	NaN
87.0	NaN	NaN	gmail.com	NaN
87.0	287.0	NaN	outlook.com	NaN
87.0	NaN	NaN	yahoo.com	NaN
87.0	NaN	NaN	gmail.com	NaN

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13
1.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	2.0	0.0	1.0
1.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0
1.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0
2.0	5.0	0.0	0.0	0.0	4.0	0.0	0.0	1.0	0.0	1.0	0.0	25.0
1.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0	1.0

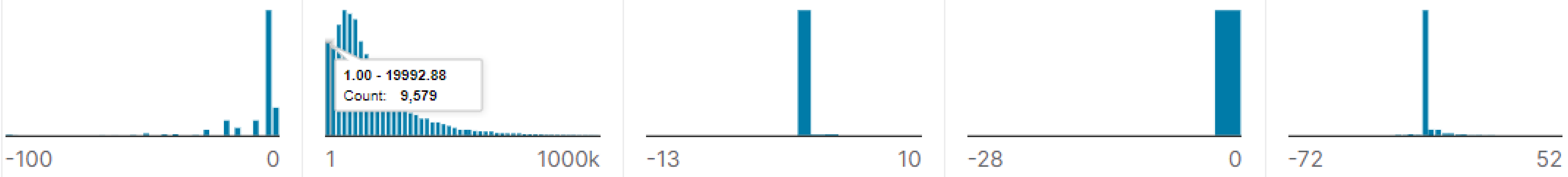
#01 대회 소개 IEEE-CIS Fraud Detection

데이터 셋: train_identity.csv

```
train_identity.info()
```

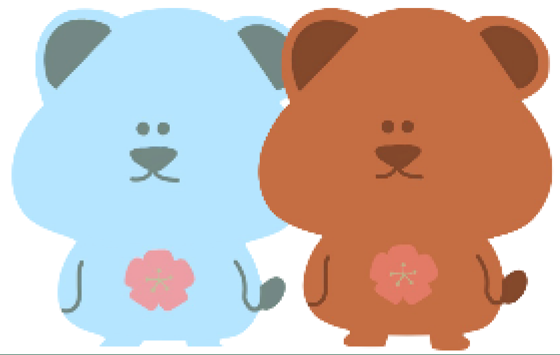
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 144233 entries, 0 to 144232
Data columns (total 41 columns):
TransactionID    144233 non-null int64
id_01            144233 non-null float64
id_02            140872 non-null float64
id_03            66324 non-null float64
id_04            66324 non-null float64
id_05            136865 non-null float64
id_06            136865 non-null float64
id_07            5155 non-null float64
id_08            5155 non-null float64
id_09            74926 non-null float64
id_10            74926 non-null float64
id_11            140978 non-null float64
id_12            144233 non-null object
id_13            127320 non-null float64
id_14            80044 non-null float64
id_15            140985 non-null object
id_16            129340 non-null object
id_17            139369 non-null float64
```

Detail   Compact   Column   10 of 41 columns



# id_01	# id_02	# id_03	# id_04	# id_05
0.0	70787.0			
-5.0	98945.0			0.0
-5.0	191631.0	0.0	0.0	0.0
-5.0	221832.0			0.0

Tackling Class Imbalance



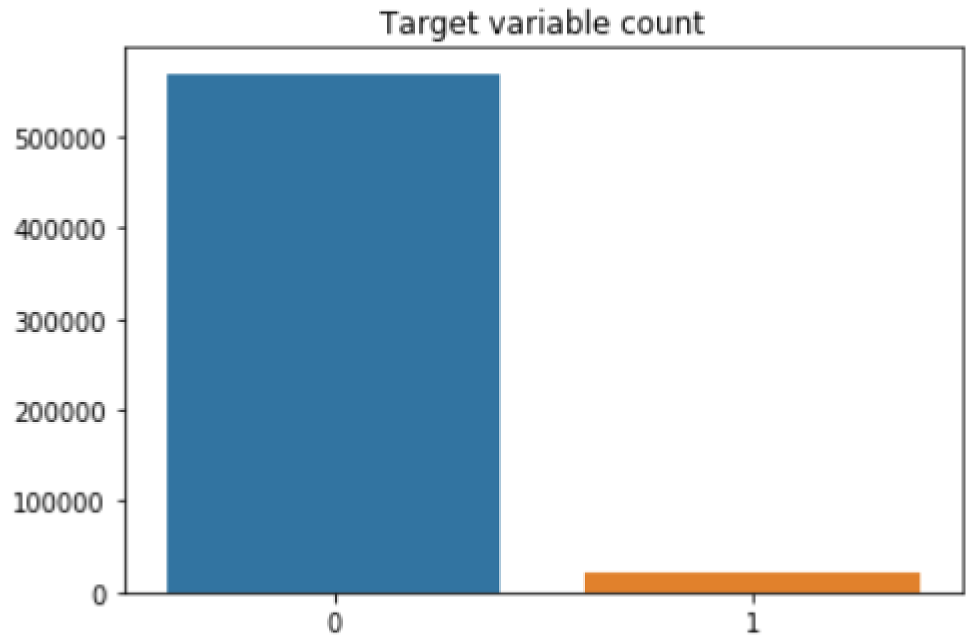
#02 Tackling Class Imbalance

Dataset Imbalance

- We will discuss about class imbalance problem which is occus often more in problems like fraudulent transaction identification and spam identification

Target variable

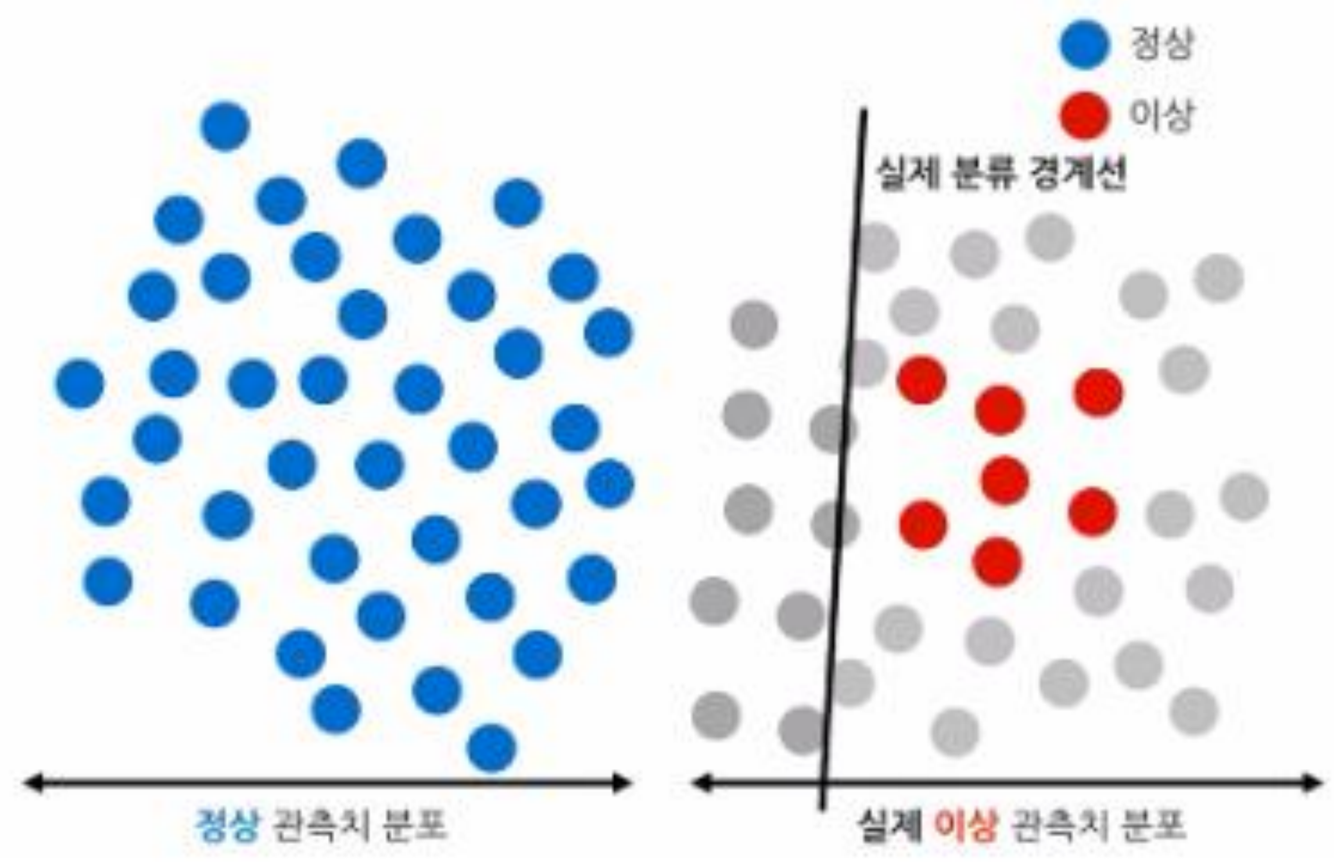
```
x=train_transactions['isFraud'].value_counts().values
sns.barplot([0,1],x)
plt.title('Target variable count')
```

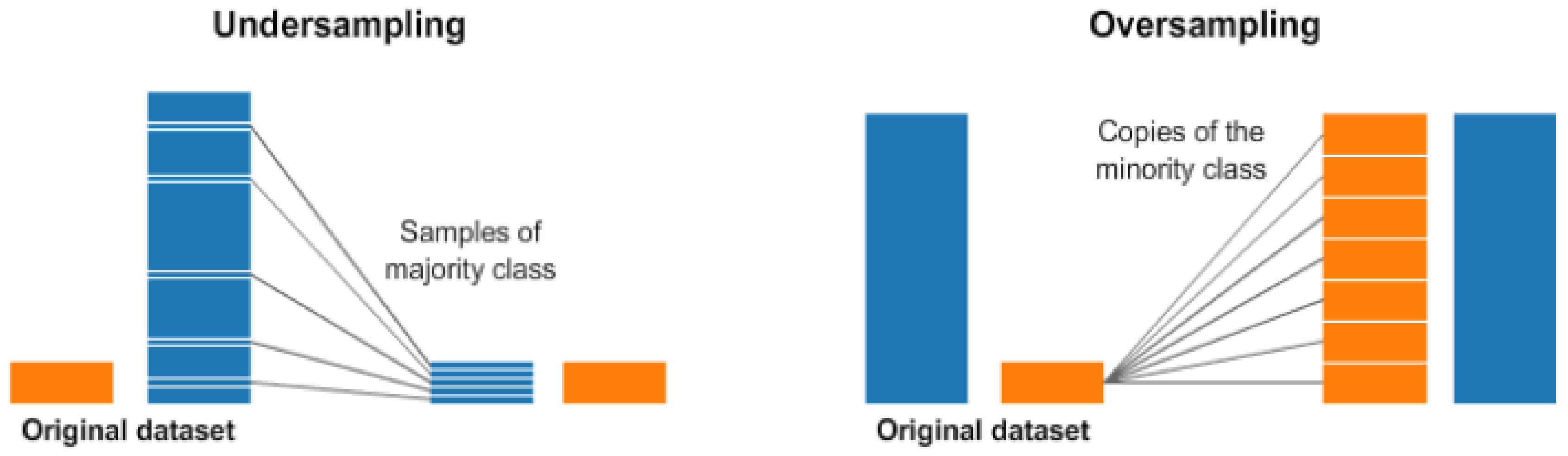


- There is clearly a class imbalance problem.
- 보통 데이터 양이 적은 이상 데이터가 target 값이 되는 경우가 많다 (사기 거래 < 정상 거래, 암 환자< 암에 걸리지 않은 환자)
- 정상 데이터를 정확히 분류하는 것보다, 이상 데이터를 정확이 분류하는 것이 더 중요하다.

#02 Tackling Class Imbalance

Resampling: solving dataset imbalance problem





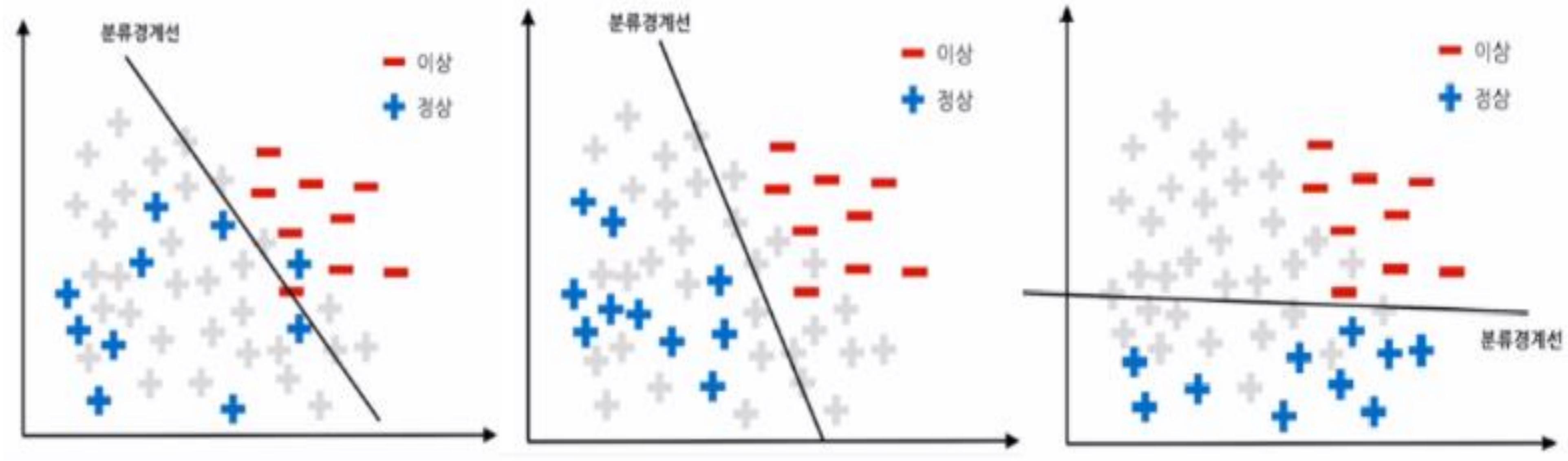
- A widely adopted technique for dealing with highly unbalanced datasets is called resampling.

➢ Under sampling: removing samples from the majority class
➢ Oversampling: adding more examples from the minority class
➢ Sampling 적용 전 PCA, T-SNE등의 Dimension reduction 기법 적용

#02 Tackling Class Imbalance

Under-sampling

1. Random under sampling




```
from imblearn.under_sampling import RandomUnderSampler

ran=RandomUnderSampler(return_indices=True) ##intialize 
X_rs,y_rs,dropped = ran.fit_sample(X,y)

print("The number of removed indices are ",len(dropped))
plot_2d_space(X_rs,y_rs,X,y,'Random under sampling')
```

- imblearn(imbalance learning)모듈을 이용해 불균형 데이터를 처리할 수 있다.

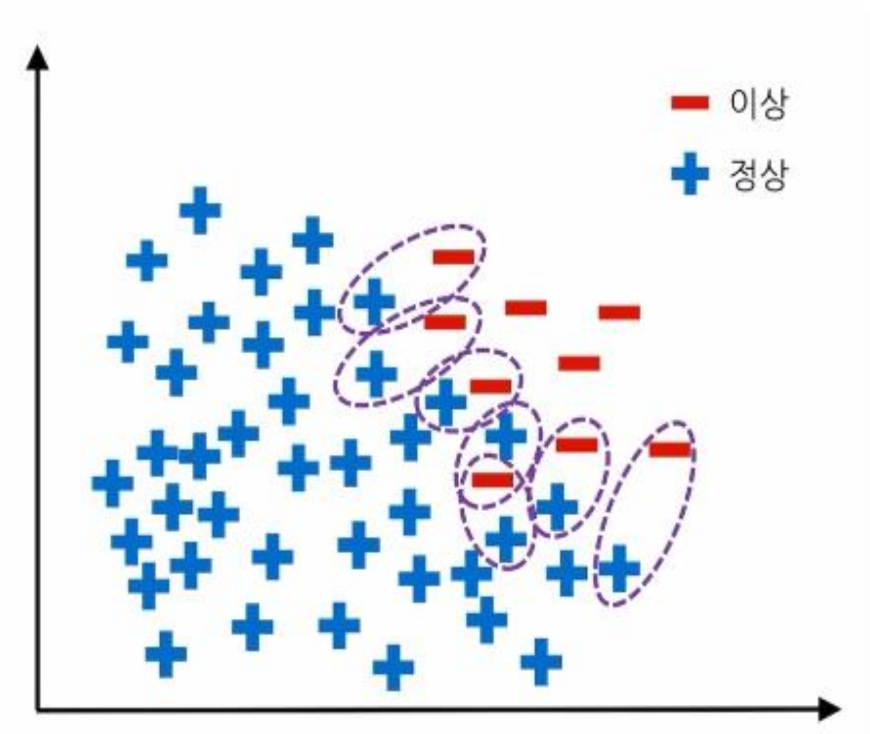
- Random sampling 이므로 시행 시마다 다른 결과를 낳는다.
- 샘플링 시행마다 모델의 성능이 달라지는 문제.

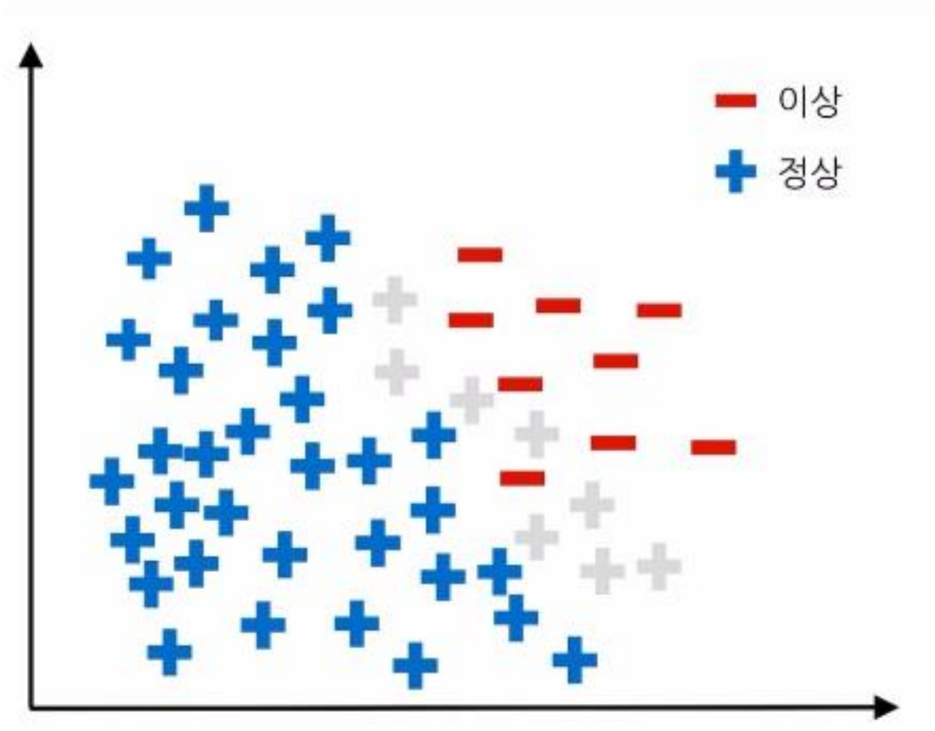
#02 Tackling Class Imbalance

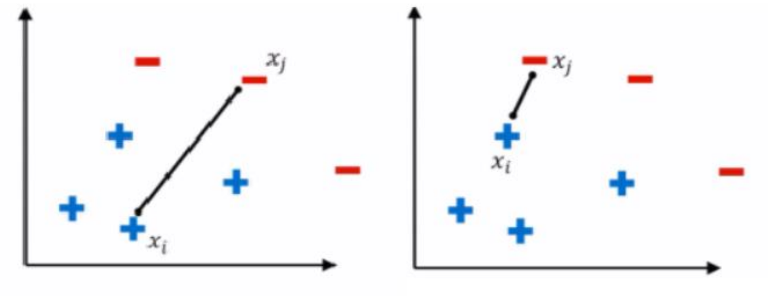
Under-sampling

2. Tomek Links

- 서로 다른 클래스의 데이터 두 점을 연결해 그 거리가 짧은 Tomek Link를 찾는다.
- 모든 tomek link를 색출하고, 그 중 다수의 클래스(정상치)에 해당하는 데이터를 삭제한다.
- Results





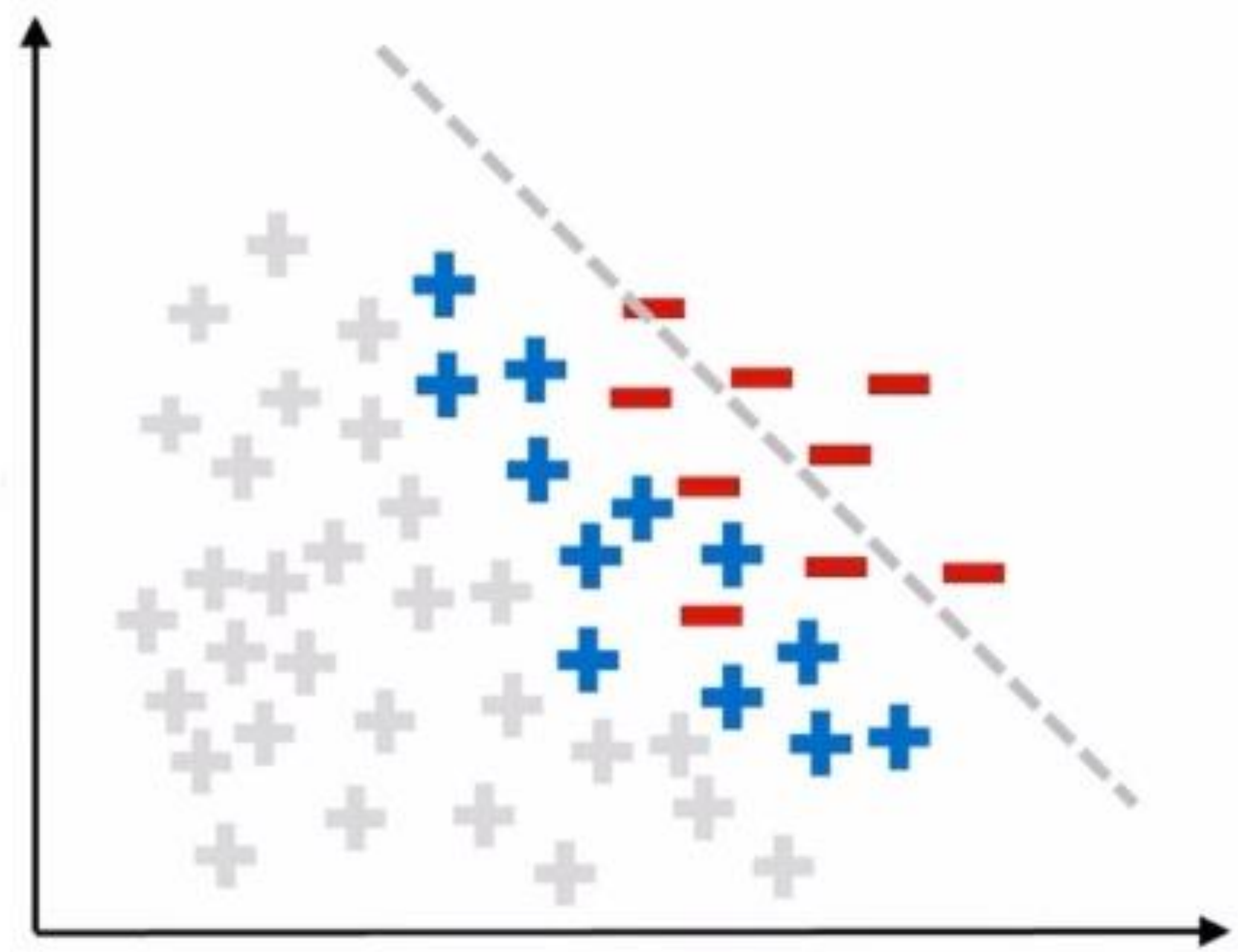


#02 Tackling Class Imbalance

Under-sampling

3. Condensed Nearest Neighbor(cnn) Rule

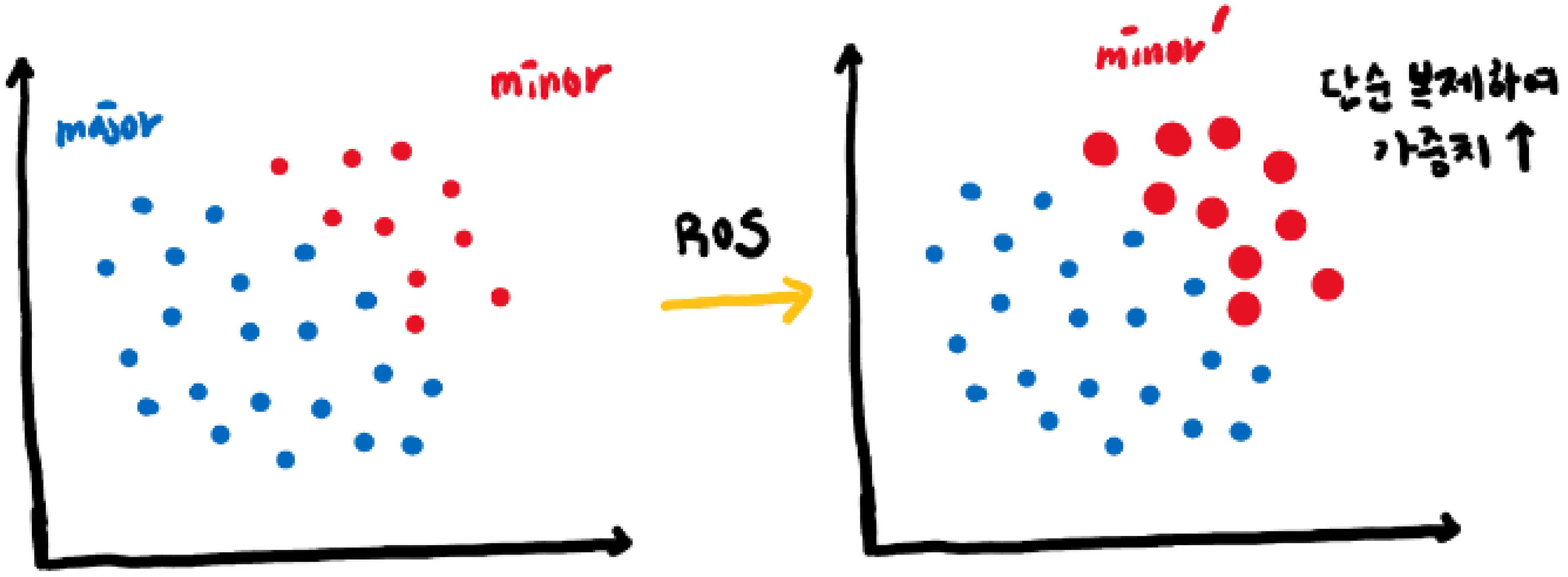
- K=1인 k-NN(1-NN)을 이용
- 정상치(다수 클래스)와 이상치(소수 클래스)에서 각각 하나씩 데이터를 무작위 추출한 sub-dataset을 구성한다
- 정상치 데이터 값을 하나씩 sub-datset과의 1-NN을 따져본다.
- 정상치와 가까웠던 값들을 under sampling 해 준다.



#02 Tackling Class Imbalance

Over-sampling

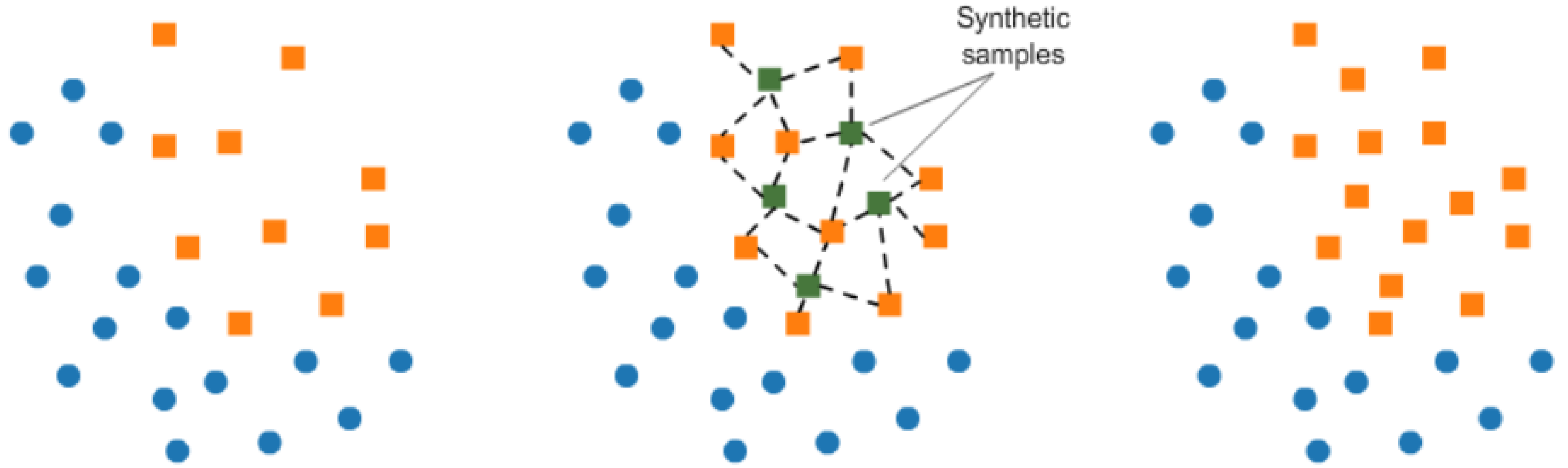
1. Resampling



➢ 기존에 존재하는 소수의 클래스를 단순히 복제하여 비율을 맞춰주는 것.
➢ 똑같은 데이터가 증식되다 보니 오버피팅의 위험

#02 Tackling Class Imbalance

Over-sampling
2. SMOTE(Synthetic Minority Oversampling Technique)




```python
from imblearn.over_sampling import SMOTE

smote = SMOTE(ratio='minority')
X_sm, y_sm = smote.fit_sample(X, y)

plot_2d_space(X_sm, y_sm,X,y, 'SMOTE over-sampling')
```

- SMOTE (Synthetic Minority Oversampling TEchnique) consists of synthesizing elements for the minority class, based on those that already exist. It works randomly picingk a point from the minority class and computing the k-nearest neighbors for this point. The synthetic points are added between the chosen point and its neighbors.
- 사전에 정한 k개의 이상치 데이터에 synthetic 공식을 적용해 가상의 데이터를 계산해 낸다.

X의 nearest neighbor

$$\text{Sythetic} = X + \underbrace{u}_{\text{균등분포}(0,1)} \cdot (X_{(nn)} - X)$$

소수클래스 관찰치

#02 Tackling Class Imbalance

Over-sampling

3. Borderline - SMOTE

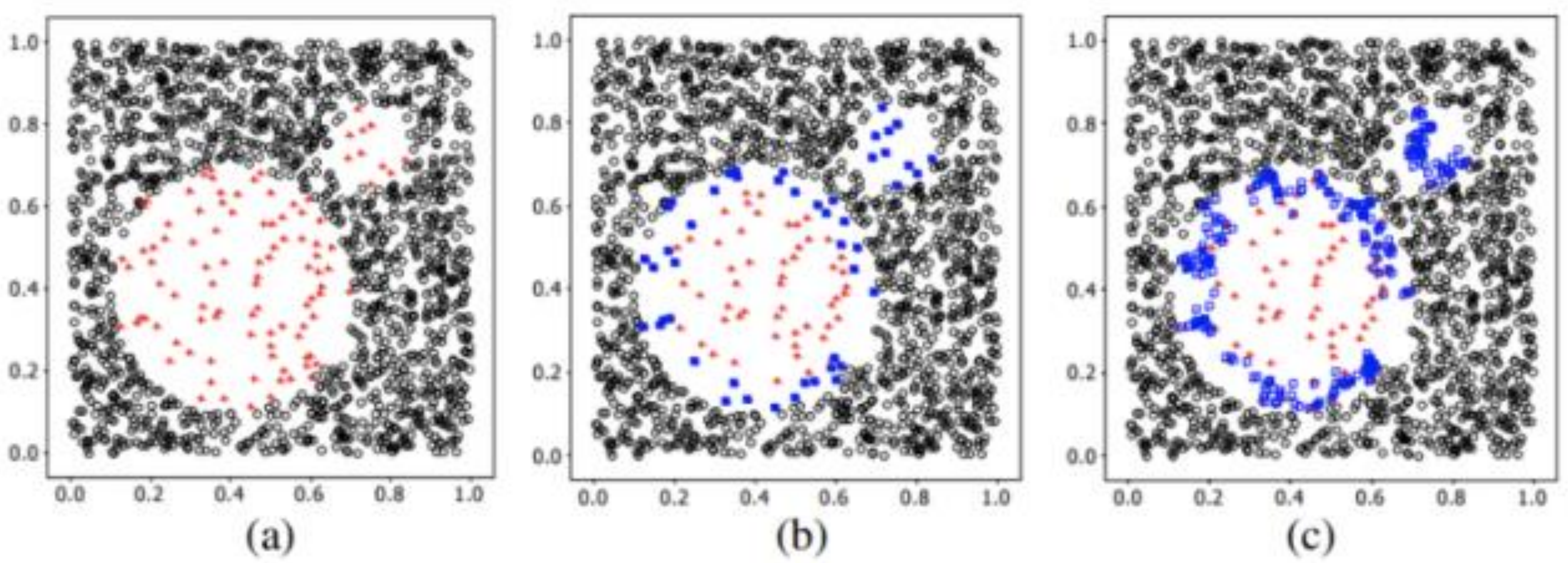


Fig. 1. (a) The original distribution of Circle data set. (b) The borderline minority examples (*solid squares*). (c) The borderline synthetic minority examples (*hollow squares*).

1. k = k' : Noise 관측치
2. k/2 < k' < k : Danger 관측치
3. 0 =< k' =< k/2 : Safe 관측치

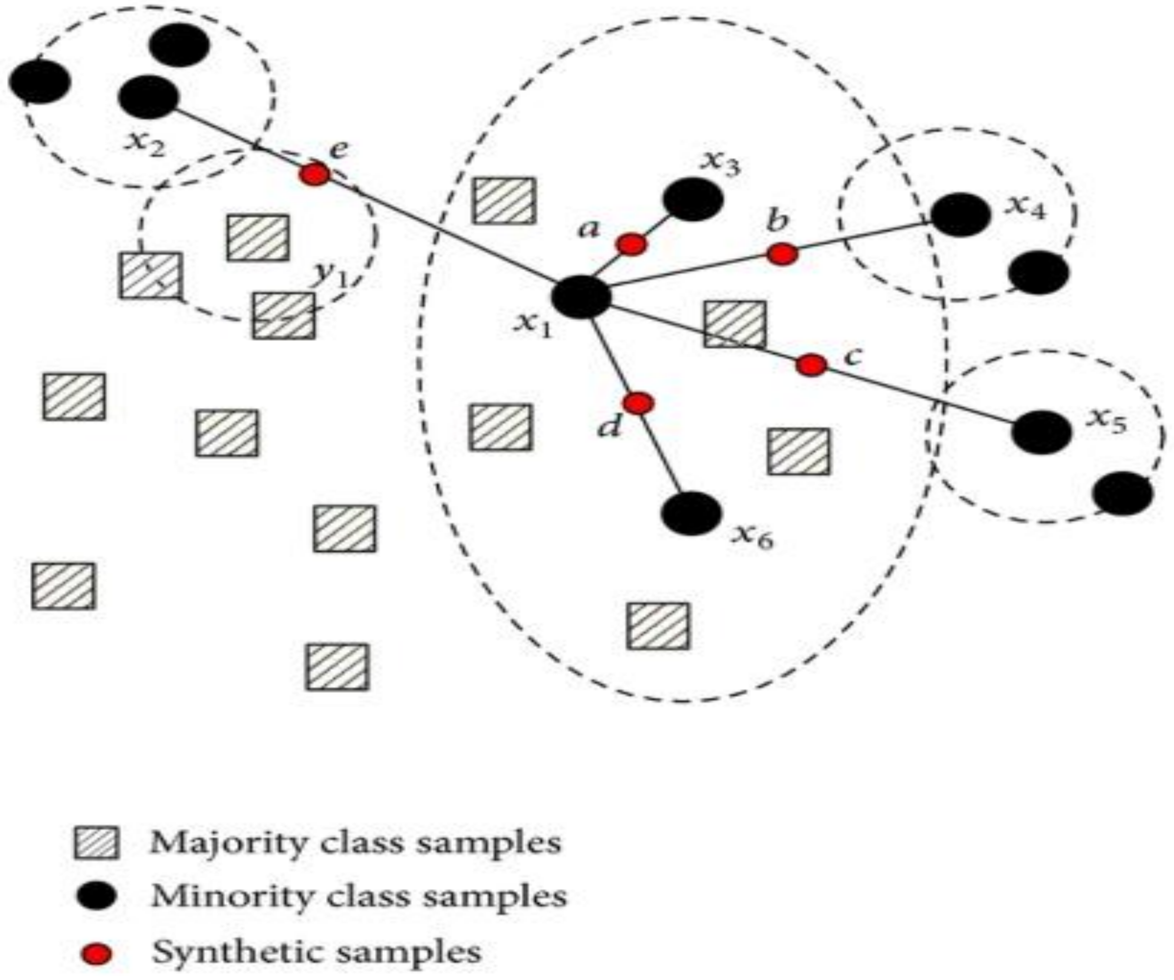
- 경계 값(borderline)으로 판별된 이상치 데이터에만 SMOTE oversampling을 적용하는 방법
- K개의 nearest neighbor를 찾는다. 그 neighbor 관측치들의 클래스에 따라 borderline 인지 평가한다.

#02 Tackling Class Imbalance

Over-sampling

4. ADASYN(Adaptive Synthetic Sampling Approach)

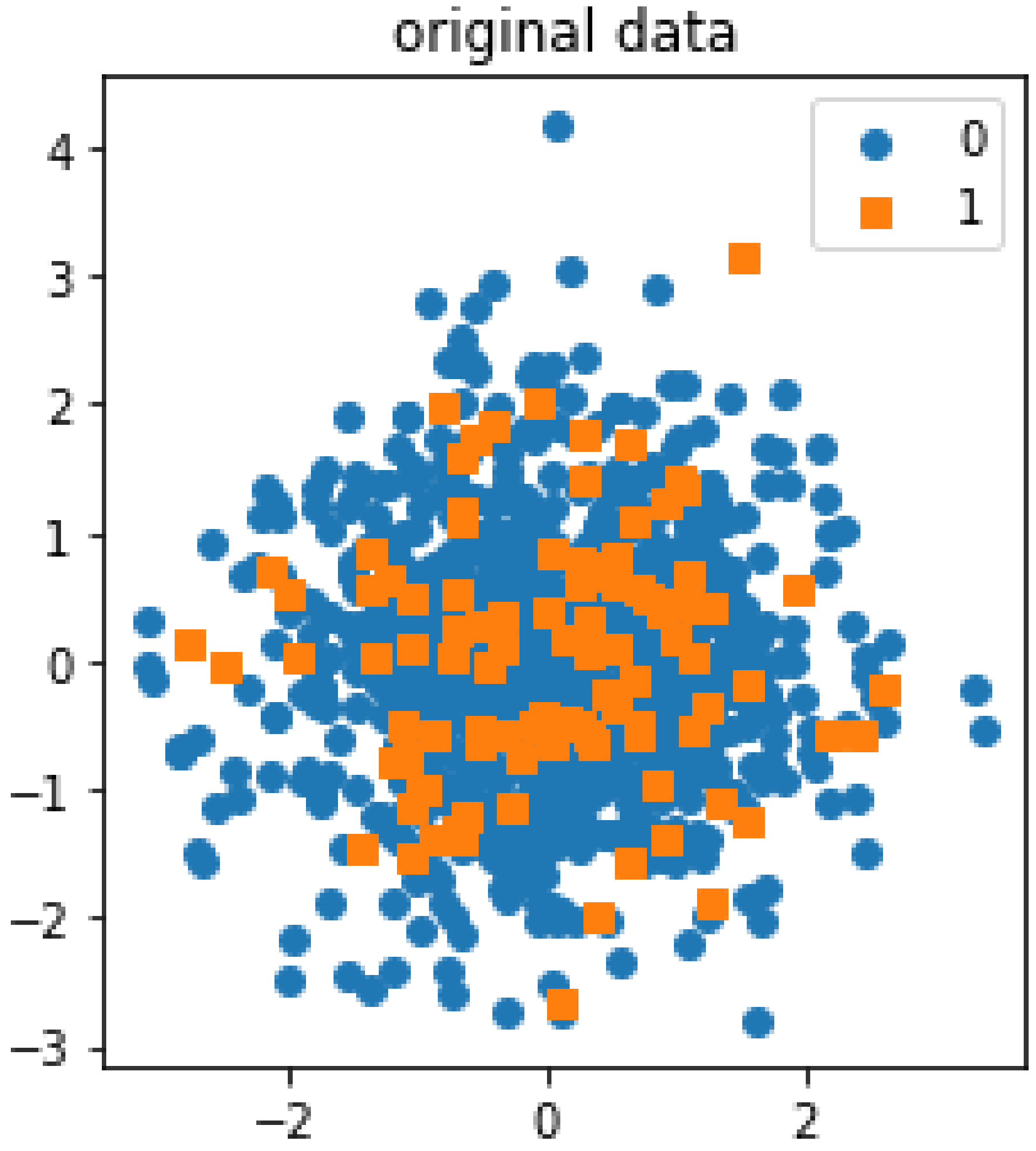
- Borderline-SMOTE와 유사하지만, 데이터의 위치에 따라 유동적으로 sampling 개수(k)를 다르게 설정한다.
- 모든 소수 클래스 데이터 각각에 대해 k개의 주변 데이터를 탐색하고, 그중 다수(정상치)클래스 비율을 계산한다. (R_i)
- R_i * (정상치 클래스 개수 – 소수 클래스 개수)를 곱해주고 반올림한다. 이렇게 얻은 정수 값 만큼 SMOTE를 적용해 oversampling 해 준다.

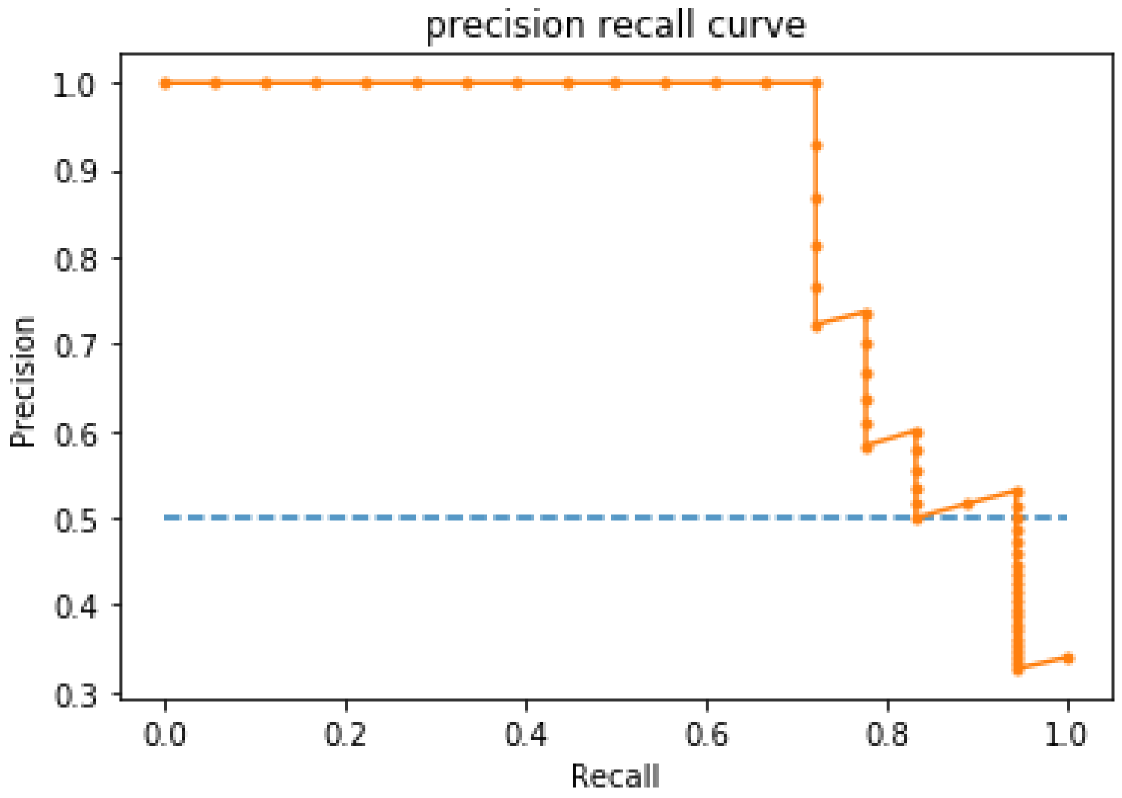


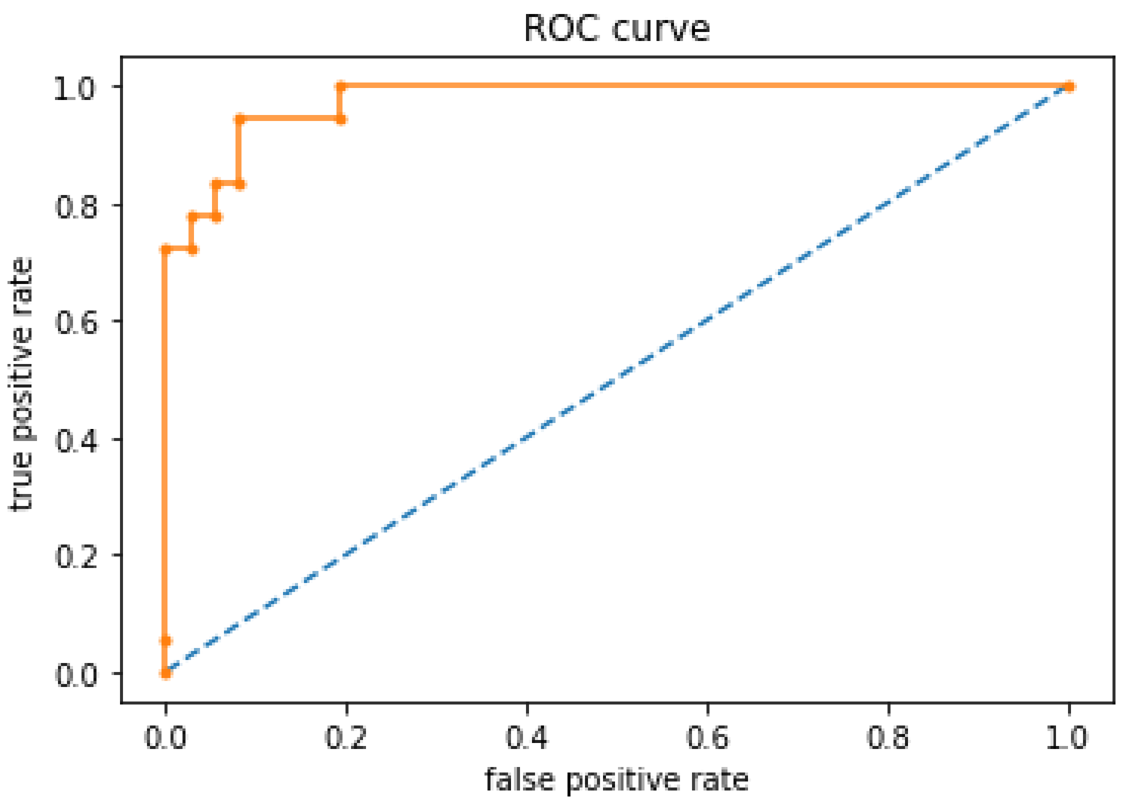
Result

#03 Result

- **Original data**

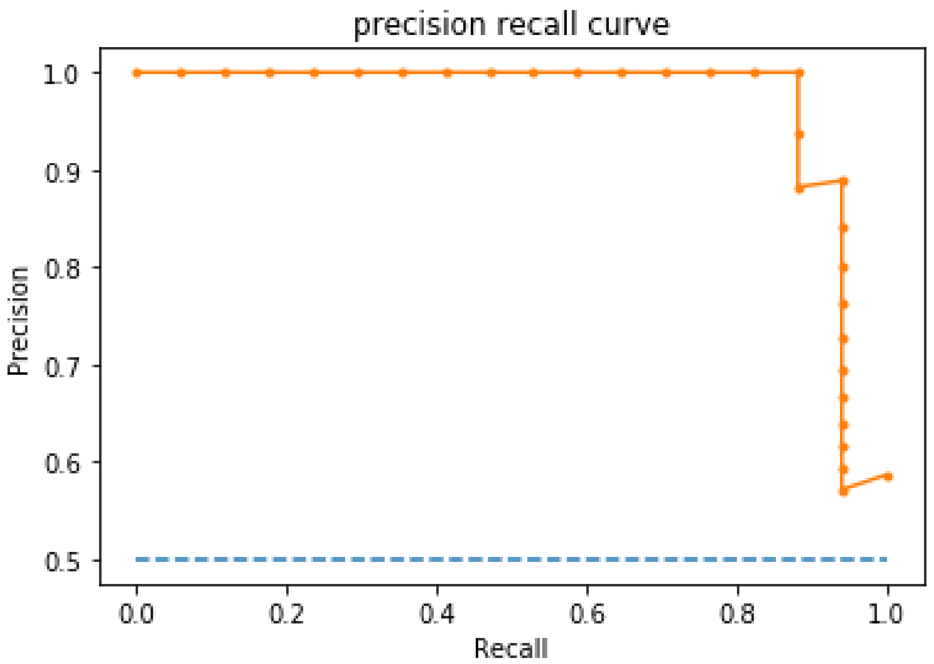


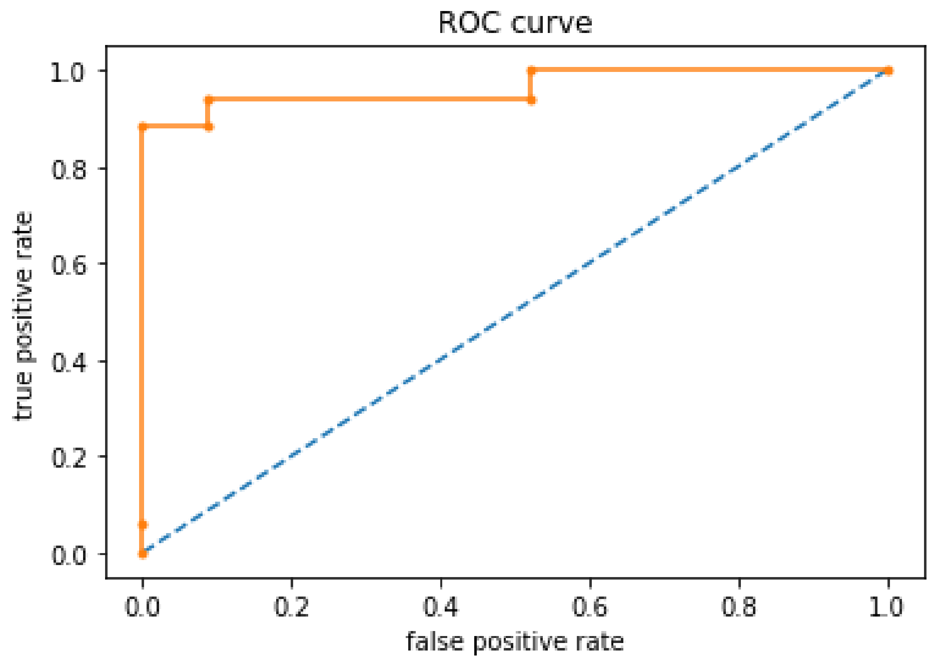


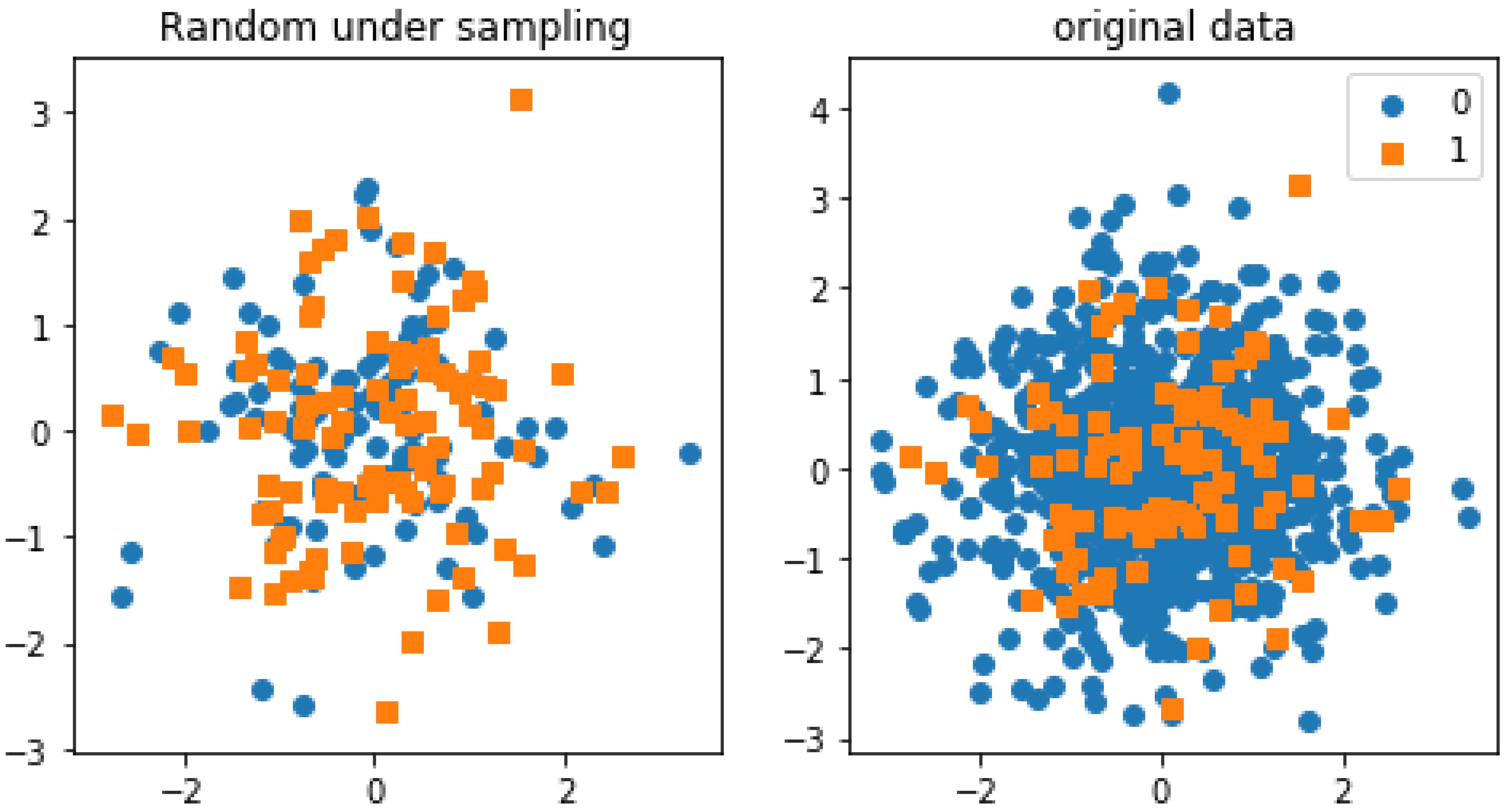


#03 Result

- **Random under-sampling with imbalanced-learn**
  - The number of removed indices are 200



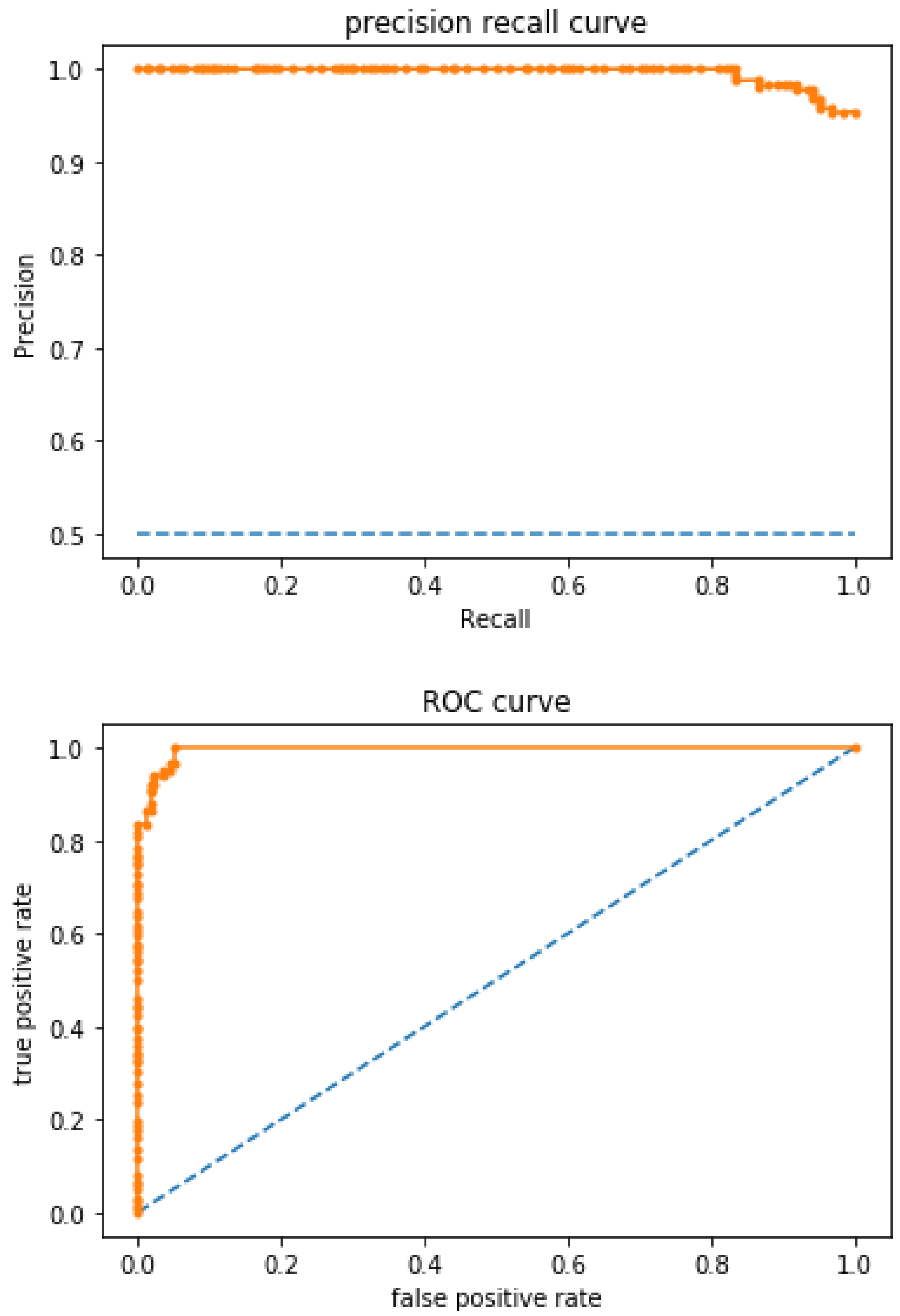


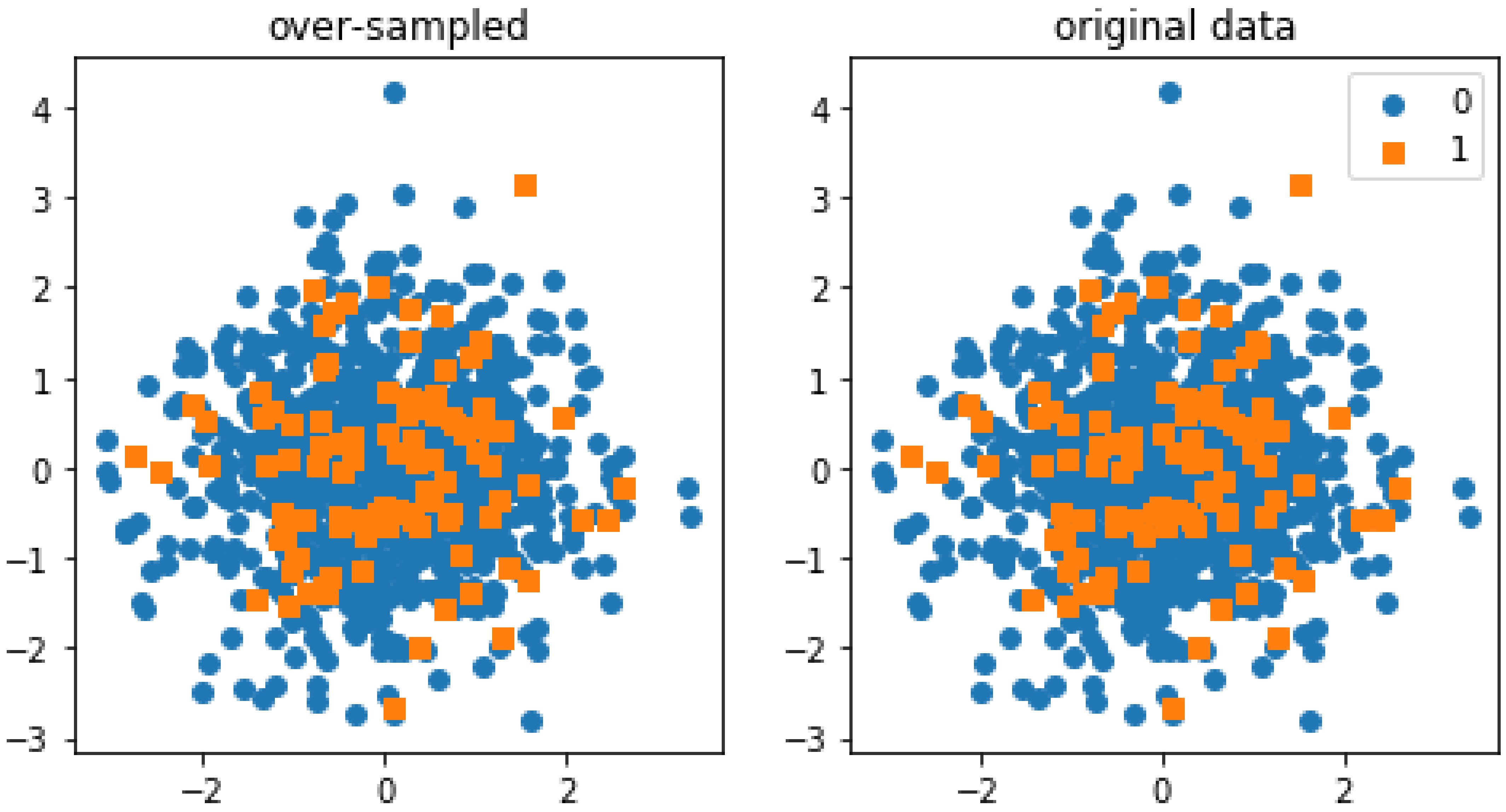


#03 Result

▪ Random over-sampling with imbalanced-learn

- The new data contains 1800 rows

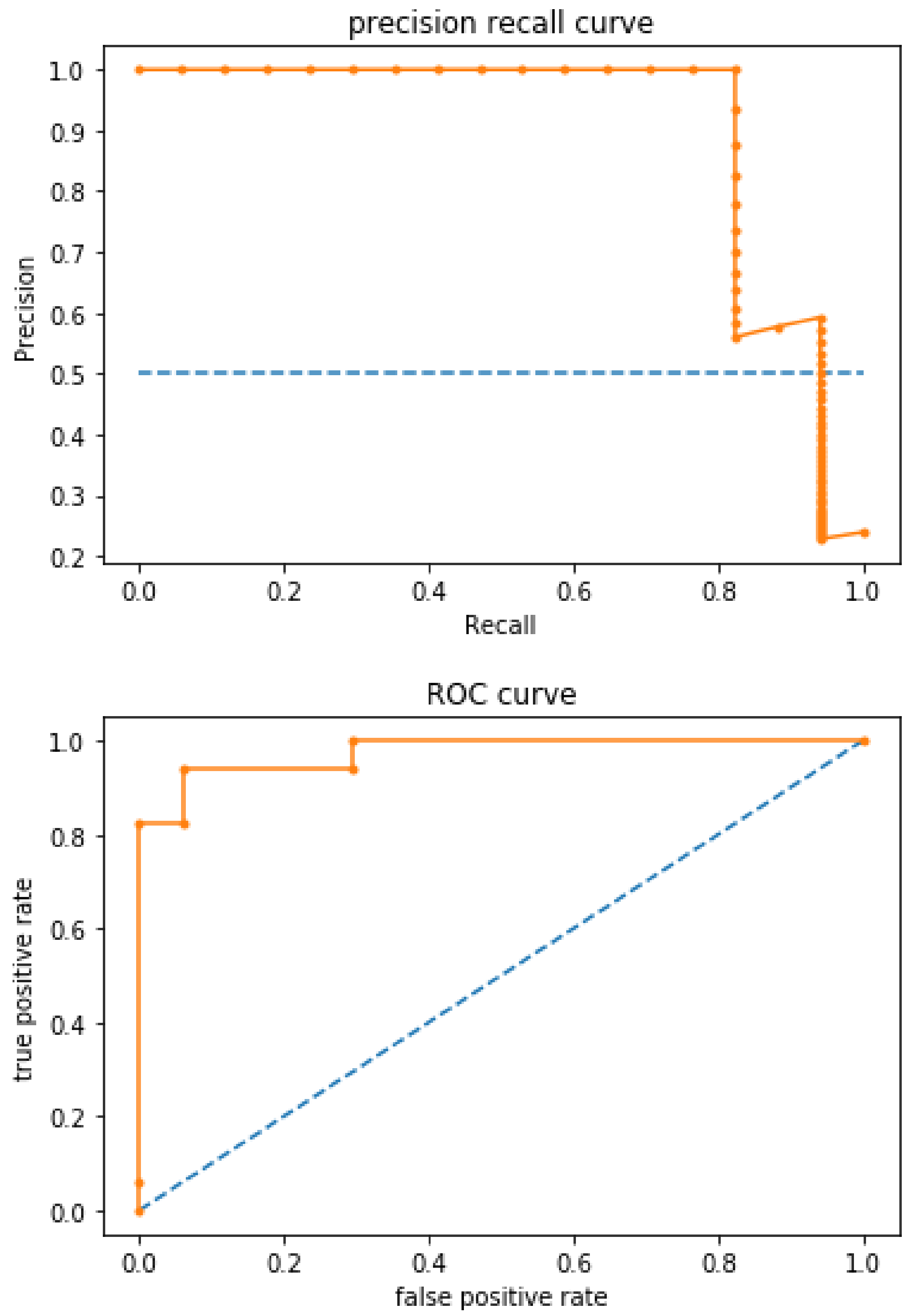


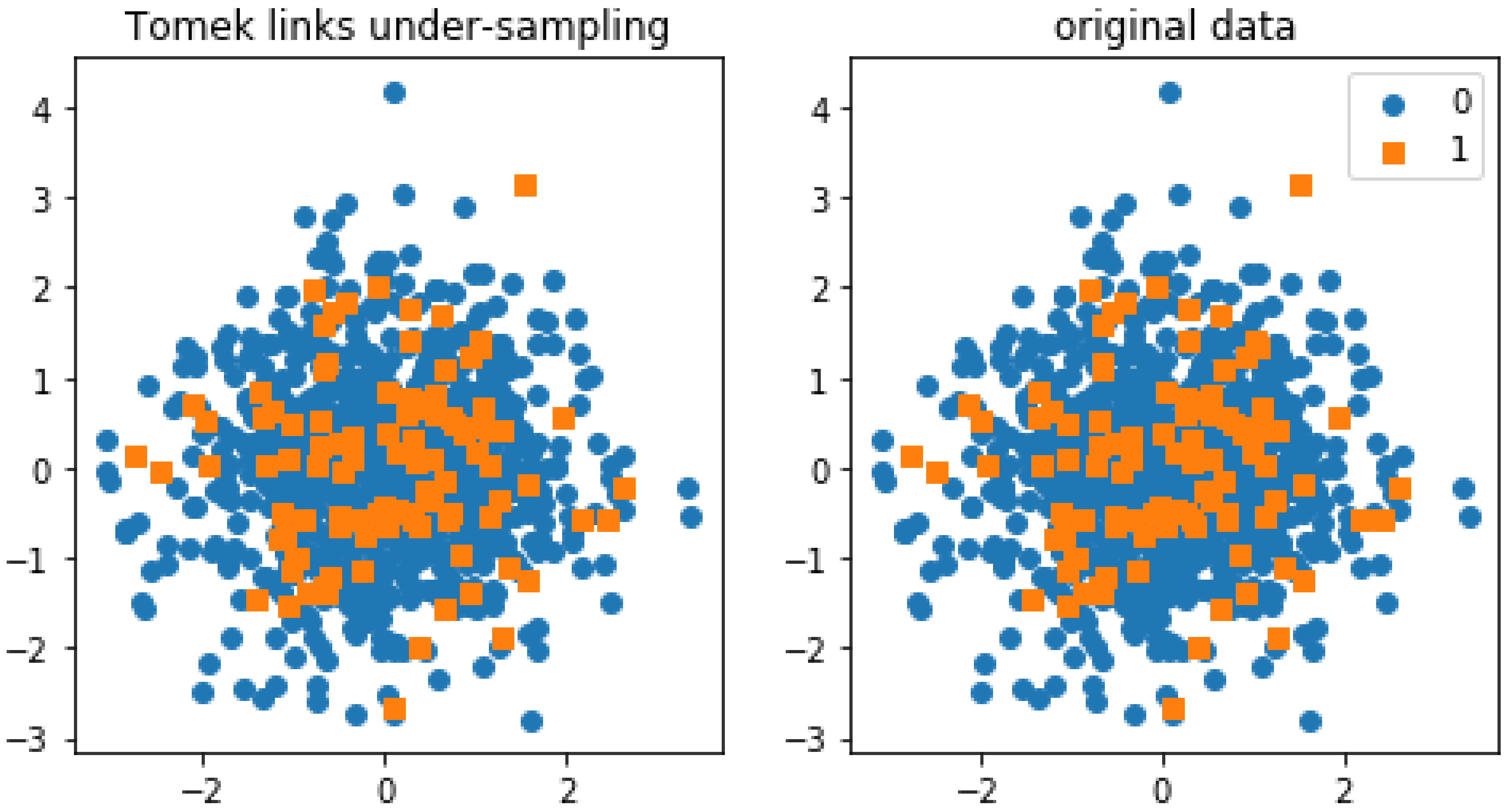


#03 Result

▪ Tomek links : Under-sampling

- The number of removed indexes are 996

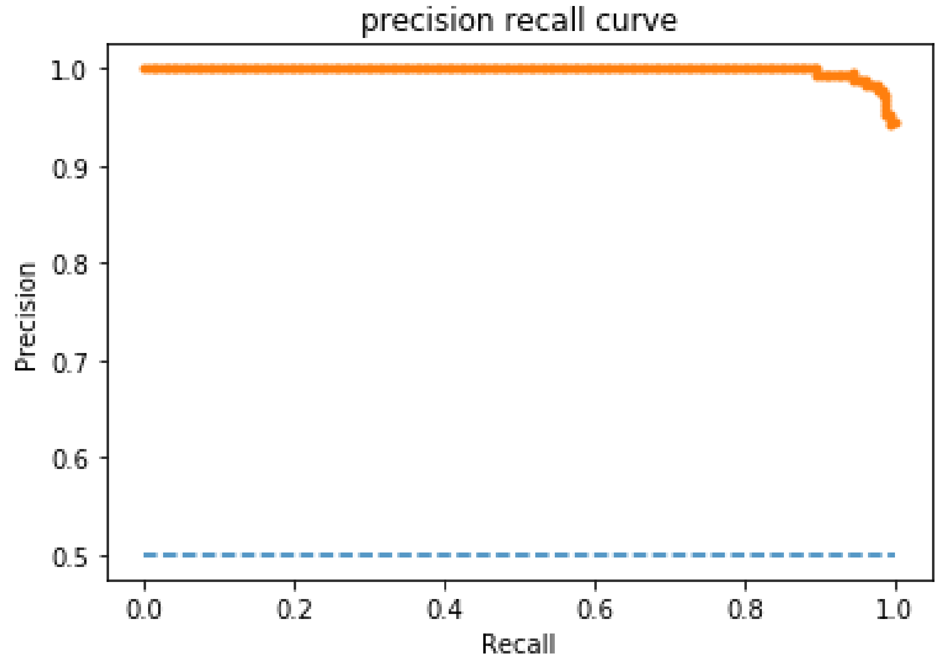


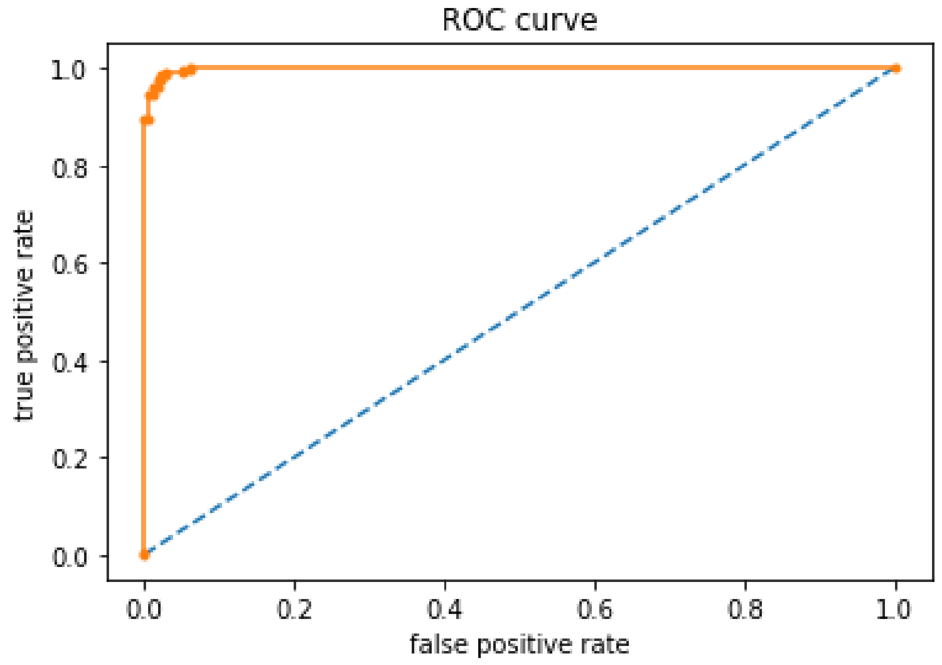


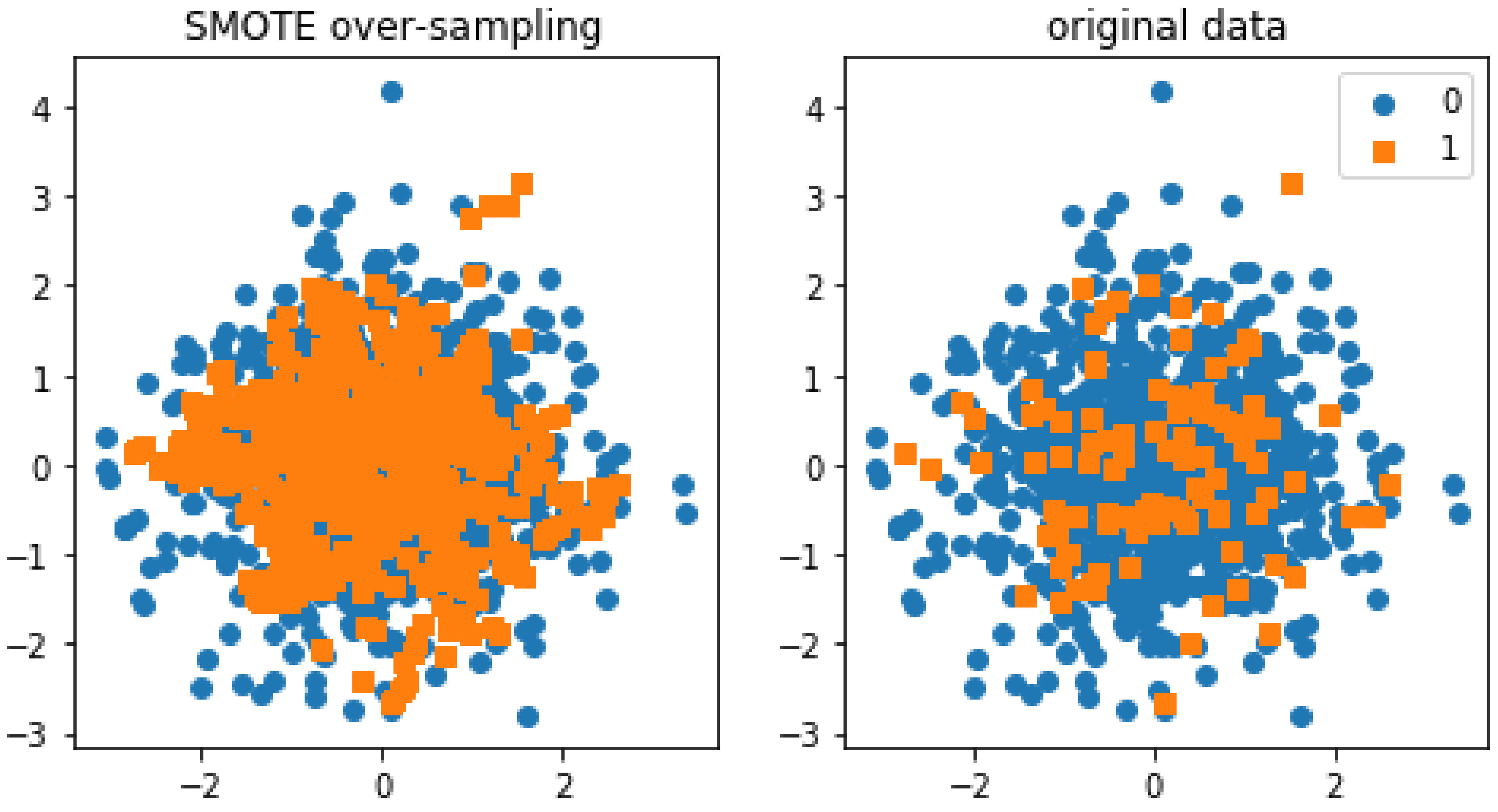
#03 Result

▪ SMOTE : Over-sampling

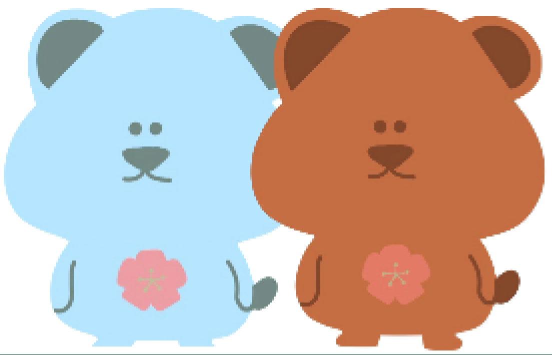
- The new data contains 1800 rows





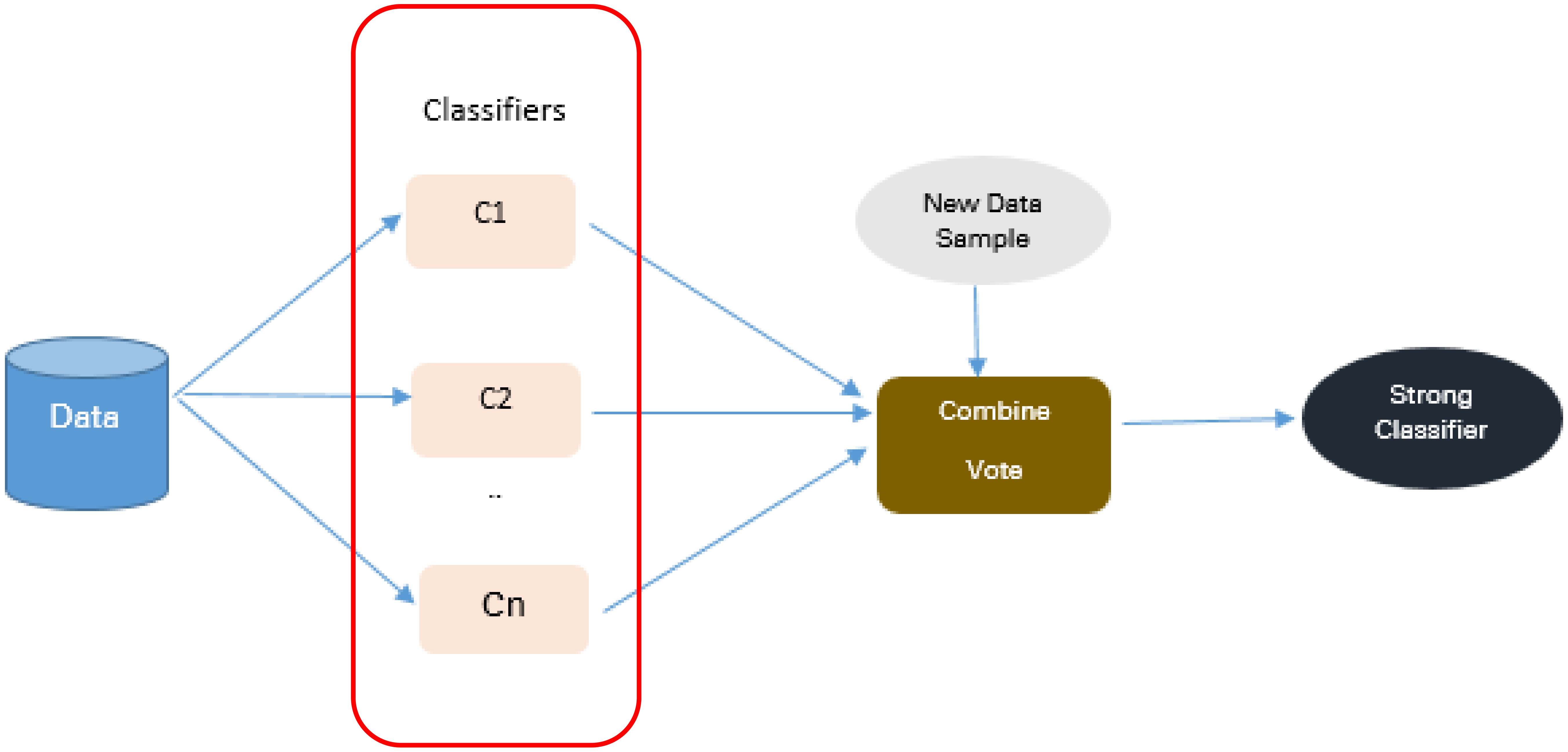


Algorithmic Ensemble techniques



#04 Algorithmic Ensemble techniques

- **Ensemble techniques**



#04 Algorithmic Ensemble techniques

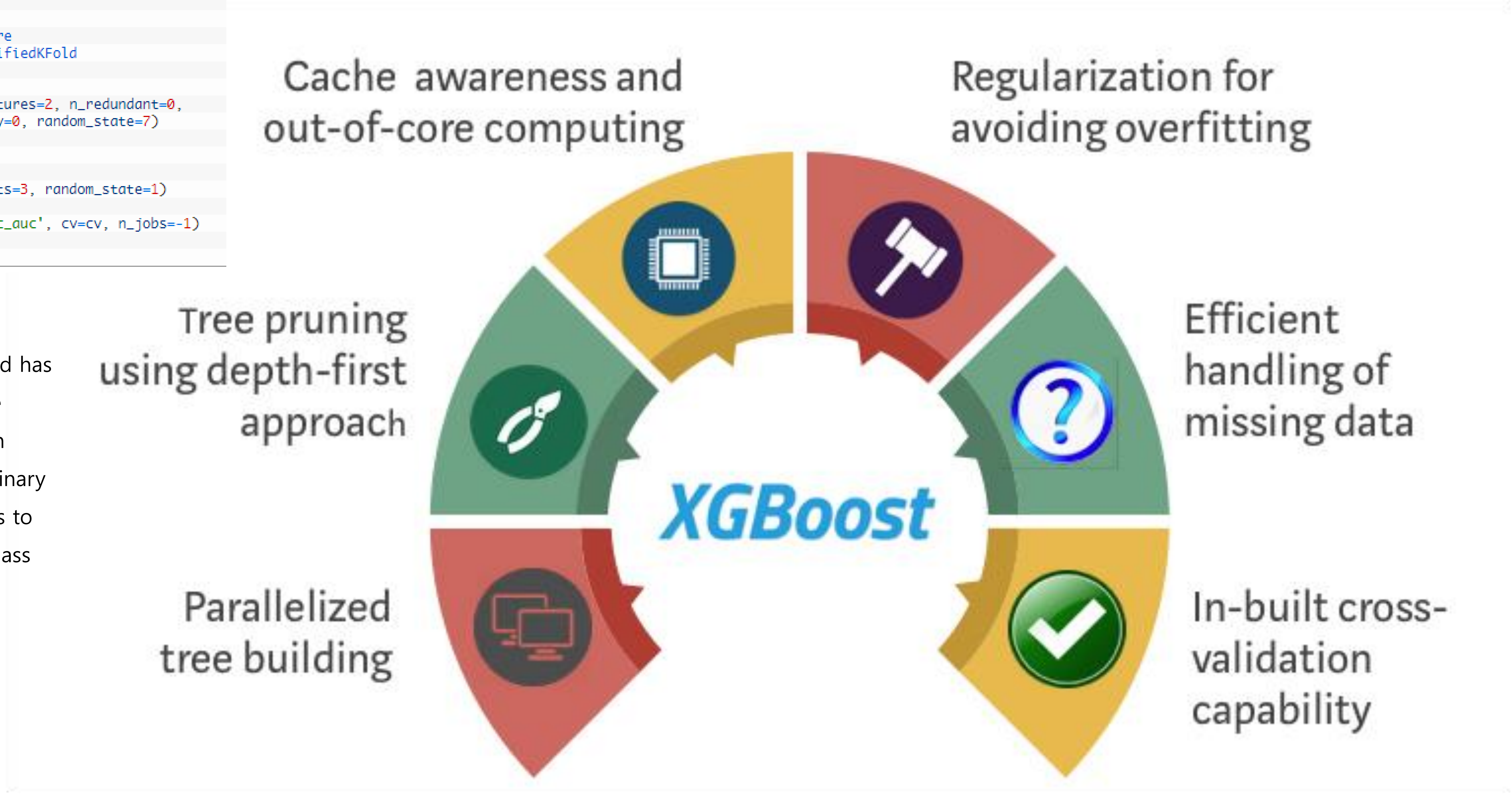
▪ XGBoost

(Class Weighted XGBoost or Cost-Sensitive XGBoost)

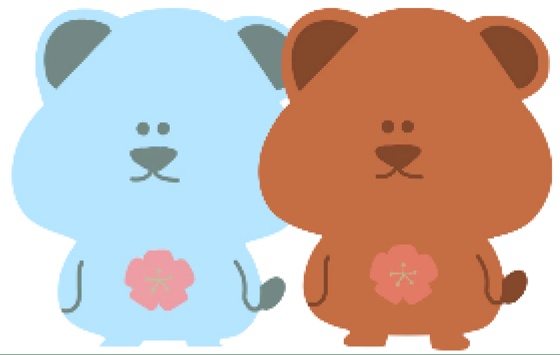
```python
# fit xgboost on an imbalanced classification dataset
from numpy import mean
from sklearn.datasets import make_classification
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold
from xgboost import XGBClassifier
# generate dataset
X, y = make_classification(n_samples=10000, n_features=2, n_redundant=0,
    n_clusters_per_class=2, weights=[0.99], flip_y=0, random_state=7)
# define model
model = XGBClassifier()
# define evaluation procedure
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
# evaluate model
scores = cross_val_score(model, X, y, scoring='roc_auc', cv=cv, n_jobs=-1)
# summarize performance
print('Mean ROC AUC: %.5f' % mean(scores))
```

- scale_pos_weight

: hyperparameter is set to the value of 1.0 and has the effect of weighing the balance of positive examples, relative to negative examples when boosting decision trees. For an imbalanced binary classification dataset, the negative class refers to the majority class (class 0) and the positive class refers to the minority class (class 1).



Resampling technique vs. ensemble method



#05 Resampling technique vs. ensemble method

- A Comparison of Ensemble Methods Combining Resampling Techniques for Class Imbalanced Data(논문)

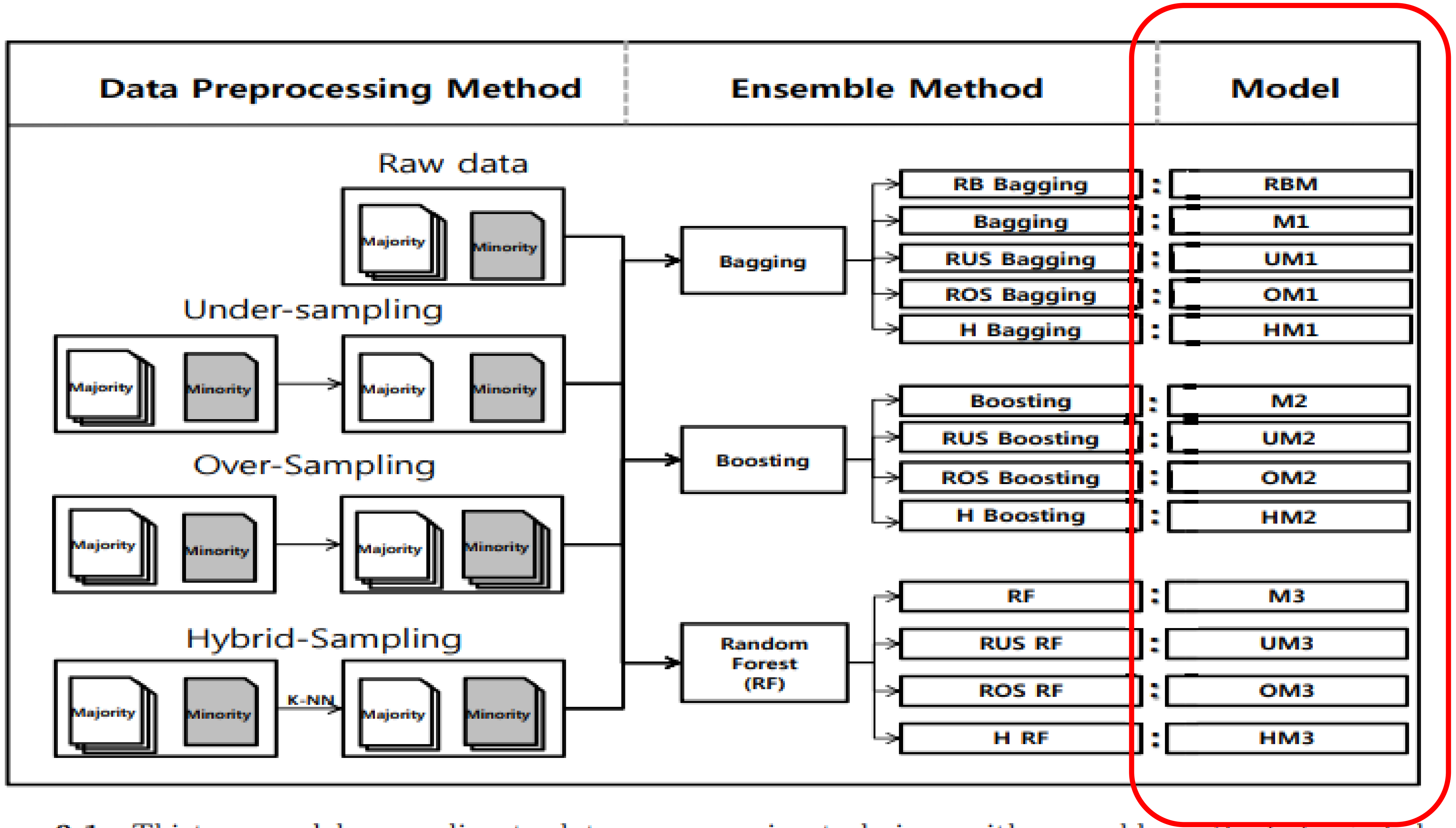


Figure 3.1. Thirteen models according to data preprocessing technique with ensemble methods for imbalanced data

- 13가지의 모형을 *불균형 데이터에 적합 (7:3 분할, 100번 수행)
- 앙상블을 사용할 때 예측모형의 수를 100개로 똑같이 고정

*UCI repository (Newman 등, 1998)와 NASA Metrics Data Program (Sayyad와 Menzies, 2005)로부터 소수 집단의 비율이 10% 이하인 실제 데이터 10개 선정

#05 Resampling technique vs. ensemble method

▪ A Comparison of Ensemble Methods Combining Resampling Techniques for Class Imbalanced Data(논문)

Table 5.2. AUC results for 10 data set using thirteen models

	Ecoli	Page-Block	Abalone	Flag	CM1	Vowel	Cleveland	Hyper-thyroid	Letter	PC1
RBM	0.816±0.002	0.942±0.002	0.806±0.002	0.784±0.004	0.709±0.003	0.985±0.002	0.806±0.003	0.981±0.004	0.967±0.003	0.813±0.006
M1	0.735±0.012	0.924±0.020	0.653±0.022	0.544±0.021	0.694±0.035	0.998±0.034	0.595±0.033	0.979±0.035	0.975±0.034	0.675±0.025
M2	0.731±0.034	0.921±0.026	0.722±0.022	0.567±0.035	0.694±0.025	0.999±0.033	0.721±0.037	0.975±0.038	0.989±0.035	0.606±0.036
M3	0.784±0.003	0.938±0.007	0.638±0.025	0.586±0.004	0.739±0.026	0.995±0.003	0.615±0.005	0.972±0.004	0.974±0.003	0.653±0.028
UM1	0.869±0.048	0.957±0.045	0.763±0.044	0.787±0.038	0.739±0.035	0.985±0.042	0.675±0.043	0.978±0.041	0.996±0.045	0.843±0.038
UM2	0.907±0.035	0.964±0.034	0.797±0.037	0.696±0.036	0.718±0.032	0.976±0.037	0.582±0.038	0.984±0.036	0.978±0.036	0.812±0.037
UM3	0.812±0.045	0.968±0.048	0.800±0.042	0.678±0.041	0.745±0.028	0.989±0.043	0.812±0.040	0.980±0.042	0.981±0.042	0.824±0.041
OM1	0.806±0.038	0.925±0.035	0.718±0.024	0.725±0.030	0.649±0.015	0.945±0.031	0.712±0.033	0.991±0.035	0.997±0.032	0.715±0.036
OM2	0.773±0.011	0.961±0.026	0.815±0.017	0.717±0.015	0.687±0.010	0.998±0.023	0.832±0.027	0.989±0.025	0.987±0.017	0.831±0.029
OM3	0.685±0.067	0.956±0.069	0.670±0.036	0.865±0.068	0.723±0.028	0.962±0.065	0.745±0.061	0.981±0.059	0.987±0.063	0.725±0.067
HM1	0.831±0.045	0.985±0.048	0.781±0.041	0.856±0.041	0.731±0.043	0.990±0.044	0.795±0.045	0.985±0.040	0.988±0.048	0.842±0.043
HM2	0.884±0.028	0.989±0.030	0.784±0.026	0.892±0.027	0.745±0.029	0.991±0.024	0.815±0.023	0.991±0.021	0.980±0.020	0.815±0.026
HM3	0.893±0.037	0.990±0.040	0.825±0.038	0.918±0.036	0.755±0.032	0.992±0.037	0.843±0.033	0.995±0.035	0.987±0.038	0.832±0.034

Table 5.3. ACC results for 10 data set using thirteen models

	Ecoli	Page-Block	Abalone	Flag	CM1	Vowel	Cleveland	Hyper-thyroid	Letter	PC1
RBM	0.814±0.005	0.942±0.003	0.699±0.003	0.593±0.004	0.718±0.002	0.939±0.003	0.808±0.002	0.961±0.002	0.954±0.003	0.694±0.005
M1	0.849±0.001	0.948±0.002	0.945±0.005	0.834±0.007	0.906±0.003	0.983±0.005	0.904±0.008	0.993±0.009	0.996±0.011	0.945±0.012
M2	0.858±0.003	0.967±0.006	0.960±0.003	0.842±0.008	0.886±0.004	0.997±0.007	0.942±0.001	0.989±0.002	0.999±0.005	0.926±0.008
M3	0.913±0.004	0.945±0.005	0.938±0.007	0.864±0.010	0.926±0.012	0.990±0.013	0.923±0.015	0.990±0.012	0.997±0.011	0.923±0.014
UM1	0.842±0.034	0.966±0.031	0.698±0.058	0.610±0.042	0.664±0.027	0.956±0.028	0.698±0.034	0.968±0.029	0.984±0.025	0.756±0.022
UM2	0.832±0.037	0.968±0.034	0.743±0.044	0.525±0.037	0.652±0.034	0.933±0.031	0.581±0.029	0.959±0.024	0.997±0.028	0.742±0.025
UM3	0.822±0.026	0.965±0.020	0.756±0.047	0.576±0.039	0.681±0.034	0.960±0.035	0.895±0.031	0.948±0.038	0.990±0.035	0.698±0.031
OM1	0.861±0.021	0.921±0.028	0.932±0.008	0.797±0.012	0.829±0.026	0.970±0.019	0.904±0.020	0.978±0.015	0.977±0.014	0.925±0.012
OM2	0.879±0.016	0.949±0.014	0.925±0.005	0.814±0.015	0.805±0.014	0.997±0.013	0.942±0.016	0.982±0.011	0.999±0.018	0.937±0.017
OM3	0.859±0.042	0.935±0.038	0.954±0.005	0.864±0.013	0.893±0.016	0.980±0.019	0.938±0.015	0.979±0.019	0.996±0.013	0.838±0.011
HM1	0.851±0.034	0.954±0.031	0.788±0.032	0.881±0.024	0.748±0.025	0.973±0.022	0.769±0.028	0.968±0.025	0.989±0.021	0.824±0.026
HM2	0.871±0.042	0.969±0.029	0.782±0.026	0.797±0.028	0.698±0.025	0.980±0.027	0.846±0.024	0.970±0.022	0.995±0.029	0.812±0.025
HM3	0.881±0.034	0.970±0.024	0.792±0.031	0.847±0.024	0.753±0.035	0.976±0.030	0.923±0.031	0.979±0.031	0.999±0.008	0.825±0.028

Table 5.4. *F*-measure results for 10 data set using thirteen models

	Ecoli	Page-Block	Abalone	Flag	CM1	Vowel	Cleveland	Hyper-thyroid	Letter	PC1
RBM	0.457±0.002	0.735±0.005	0.282±0.002	0.294±0.005	0.253±0.007	0.782±0.008	0.483±0.006	0.816±0.005	0.795±0.004	0.485±0.007
M1	0.526±0.029	0.875±0.028	0.425±0.031	0.185±0.035	0.232±0.036	0.926±0.027	0.395±0.031	0.962±0.033	0.961±0.029	0.685±0.024
M2	0.516±0.048	0.866±0.042	0.585±0.045	0.165±0.048	0.216±0.042	0.975±0.041	0.495±0.046	0.950±0.050	0.984±0.051	0.596±0.049
M3	0.632±0.068	0.892±0.051	0.375±0.071	0.226±0.058	0.352±0.060	0.965±0.062	0.462±0.068	0.953±0.066	0.968±0.061	0.671±0.067
UM1	0.529±0.035	0.779±0.034	0.265±0.037	0.316±0.038	0.262±0.035	0.816±0.036	0.325±0.039	0.816±0.032	0.862±0.037	0.513±0.038
UM2	0.541±0.038	0.824±0.031	0.301±0.033	0.152±0.035	0.235±0.037	0.794±0.039	0.235±0.034	0.807±0.038	0.971±0.036	0.425±0.034
UM3	0.471±0.027	0.775±0.028	0.264±0.033	0.205±0.030	0.245±0.029	0.842±0.027	0.685±0.026	0.793±0.025	0.931±0.025	0.419±0.022
OM1	0.560±0.035	0.850±0.034	0.471±0.044	0.261±0.038	0.185±0.037	0.868±0.036	0.516±0.035	0.935±0.039	0.816±0.038	0.485±0.037
OM2	0.571±0.025	0.896±0.026	0.539±0.024	0.361±0.028	0.216±0.025	0.986±0.024	0.735±0.029	0.967±0.022	0.978±0.025	0.675±0.029
OM3	0.560±0.068	0.900±0.057	0.483±0.079	0.218±0.062	0.234±0.068	0.915±0.063	0.621±0.069	0.959±0.060	0.965±0.067	0.681±0.071
HM1	0.513±0.030	0.839±0.034	0.319±0.038	0.543±0.032	0.315±0.033	0.906±0.036	0.485±0.037	0.865±0.031	0.901±0.038	0.591±0.034
HM2	0.581±0.024	0.836±0.021	0.316±0.024	0.465±0.020	0.296±0.025	0.926±0.026	0.516±0.024	0.895±0.028	0.975±0.027	0.576±0.029
HM3	0.574±0.038	0.848±0.034	0.334±0.037	0.539±0.033	0.265±0.035	0.968±0.037	0.875±0.039	0.906±0.031	0.969±0.040	0.587±0.037

Table 5.5. *G*-mean results for 10 data set using thirteen models

	Ecoli	Page-Block	Abalone	Flag	CM1	Vowel	Cleveland	Hyper-thyroid	Letter	PC1
RBM	0.807±0.003	0.923±0.002	0.796±0.002	0.778±0.004	0.709±0.003	0.967±0.005	0.804±0.002	0.979±0.003	0.959±0.004	0.812±0.002
M1	0.728±0.035	0.922±0.031	0.558±0.038	0.534±0.031	0.562±0.032	0.991±0.019	0.598±0.011	0.974±0.018	0.969±0.020	0.621±0.033
M2	0.729±0.031	0.915±0.038	0.667±0.037	0.548±0.019	0.554±0.036	0.998±0.020	0.700±0.019	0.972±0.012	0.980±0.019	0.606±0.020
M3	0.784±0.026	0.933±0.024	0.534±0.006	0.563±0.017	0.542±0.002	0.994±0.022	0.600±0.025	0.961±0.027	0.963±0.022	0.615±0.006
UM1	0.868±0.035	0.955±0.043	0.757±0.043	0.702±0.041	0.721±0.039	0.976±0.034	0.651±0.030	0.971±0.035	0.983±0.030	0.841±0.031
UM2	0.907±0.034	0.958±0.039	0.793±0.038	0.469±0.035	0.675±0.030	0.963±0.027	0.534±0.025	0.981±0.029	0.971±0.027	0.776±0.024
UM3	0.812±0.027	0.950±0.043	0.799±0.042	0.587±0.038	0.732±0.047	0.978±0.025	0.757±0.029	0.976±0.034	0.974±0.030	0.761±0.038
OM1	0.806±0.025	0.920±0.021	0.675±0.033	0.617±0.019	0.562±0.034	0.931±0.012	0.679±0.015	0.960±0.014	0.984±0.011	0.637±0.013
OM2	0.768±0.025	0.951±0.019	0.806±0.020	0.695±0.029	0.591±0.028	0.998±0.015	0.789±0.019	0.984±0.020	0.986±0.028	0.827±0.016
OM3	0.685±0.067	0.947±0.040	0.583±0.062	0.563±0.030	0.571±0.064	0.954±0.027	0.700±0.025	0.972±0.036	0.987±0.031	0.718±0.028
HM1	0.829±0.026	0.961±0.038	0.779±0.042	0.844±0.035	0.704±0.034	0.985±0.032	0.694±0.031	0.975±0.037	0.988±0.035	0.812±0.026
HM2	0.884±0.045	0.963±0.035	0.783±0.025	0.889±0.030	0.708±0.022	0.989±0.025	0.736±0.026	0.988±0.021	0.980±0.022	0.796±0.014
HM3	0.885±0.039	0.963±0.024	0.790±0.038	0.917±0.029	0.698±0.035	0.987±0.030	0.779±0.024	0.976±0.025	0.981±0.022	0.816±0.019

Table 5.6. Results for model comparison based on Wilcoxon test

comparison	p-value			
	AUC	ACC	*F*-measure	*G*-mean
HM3 vs. RBM	0.005**	0.005**	0.005**	0.203
HM3 vs. M1	0.007**	0.575	0.646	0.009**
HM3 vs. M2	0.013*	0.314	0.646	0.013*
HM3 vs. M3	0.007**	0.051	0.575	0.007**
HM3 vs. UM1	0.022*	0.005**	0.005**	0.169
HM3 vs. UM2	0.013*	0.005**	0.007**	0.053
HM3 vs. UM3	0.005**	0.005**	0.005**	0.097
HM3 vs. OM1	0.009**	0.959	0.059	0.007**
HM3 vs. OM2	0.018*	0.192	0.646	0.575
HM3 vs. OM3	0.008**	0.314	0.760	0.009**
HM3 vs. HM1	0.025*	0.059	0.059	0.202
HM3 vs. HM2	0.005**	0.011*	0.074	0.314

$*$: p-value < 0.05; $**$: p-value < 0.01

- 배깅(특히, roughly balanced bagging), 랜덤포레스트 → under-sampling: GOOD
- 부스팅 → over-sampling: GOOD
- SMOTE: 모든 앙상블 기법에서 성능 향상 → 특히 랜덤포레스트 GOOD

THANK YOU