smote

溫翎傑

2025-09-20

Table of contents

中文

1 format

Smote 演算法通常用於處理資料及類別不平衡的問題 醫療診斷中陽性病例只有 5%，陰性病例有95%。這樣的數據會導致模型偏向於預測「陰性」，忽略了重要的少數類別。 所以傳統解決辦法

1.欠採樣（undersampling）：隨機刪除部分多數類別樣本

2.過採樣（oversampling）：可能會造成過擬合

但通常有

1.smote + tomek link:適合結構化 tabular data（數值型或類別編碼後的特徵）。少數類與多數類的樣本在邊界交疊 很多，例如：醫療診斷、欺詐檢測。

特徵維度可以中等（10–100 維），但如果太高維，鄰居關係會不穩定（curse of dimensionality）。

2.smote + ENN:有明顯噪音或錯標記 (mislabeling) 的資料集。

數值型或類別型皆可，但需要能計算「鄰居」距離，所以通常會做 one-hot encoding 或標準化。 資料邊界不只模糊，還有「孤立點」例如社會科學調查：問卷中有人亂填，導致回答異常。

2 algorithm

1.對每一筆少數類別樣本x找出最接近的k個鄰居

2.隨機選出一個鄰居x_neigh

3.對x跟x_neigh之間機取一點作為新樣本：

$$x_{new} = x + \delta \times (x[neigh] - x), \delta \sim U(0, 1)$$

4.重複這個過程，直到少數類別樣本數量達到指定比例。

3 cluster

方法	類型	主要目的	優點	缺點	適用情境

SMOTE + Tomek Link | 過採樣 + 欠採樣 | 平衡數據 + 清理邊界 | 邊界更清楚，避免多數類壓制 | 刪掉邊界樣本可能過度簡化 | 分類邊界模糊，類別交疊 |

SMOTE + ENN | 過採樣 + 欠採樣/去噪 | 去除噪音樣本 | 能刪掉噪音與錯誤標籤 | 可能刪太多，少數類不足 | 資料含有雜訊、錯標問題 |

Bootstrap → SMOTE → Monte Carlo | 重抽樣 + 模擬 | 估計效能分布 | 有信賴區間，評估更穩健 | 計算量大，不直接清理數據 | 做研究、報告，需嚴謹效能估計 |

```
import matplotlib as mpl
from matplotlib import font_manager

font_path = r"C:\Windows\Fonts\msjh.ttc"  #         Noto CJK
font_manager.fontManager.addfont(font_path)
mpl.rcParams['font.family'] = font_manager.FontProperties(fname=font_path).get_name()
mpl.rcParams['axes.unicode_minus'] = False
```

4 do smote

下載所需套件

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score ,classification_report
from sklearn.preprocessing import MinMaxScaler
from imblearn.datasets import fetch_datasets
from imblearn.over_sampling import SMOTE
from imblearn.combine import SMOTETomek, SMOTEENN
sns.set(style="white")
from sklearn.model_selection import cross_val_score, StratifiedKFold
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import make_classification
from imblearn.pipeline import Pipeline
from imblearn.over_sampling import BorderlineSMOTE
from sklearn.model_selection import cross_validate
```

5 smote

下面先使用pca降維，在用套件標準化，在分成訓練集跟測試集，下面用不同的方法，

但依定要先切愛做smote ，第二個才有做到smote，

因為先對全資料做 PCA / 縮放

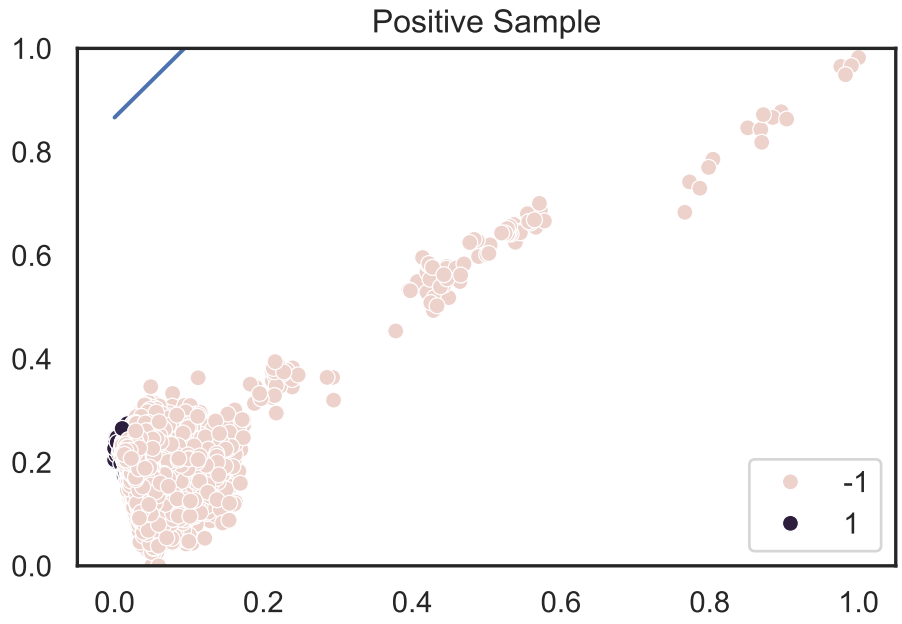
在整個 X 上 fit_transform 了 PCA 和 MinMaxScaler。

這代表 PCA 的主成分方向、縮放的最小值/最大值是「用到測試集統計量」算出來的。
模型因此「看到了」測試集的分布資訊 → 評估偏樂觀。
正確做法：只在訓練集上 fit（學到統計量/主成分），再用 transform 套到測試集。
交叉驗證時的典型外洩，把這些步驟包進 Pipeline，讓每一折只在該折的訓練子集上 fit
下面一條「為了畫 decision boundary 的 2D 視覺化流程」，
一條「為了拿到可信 AUC 的交叉驗證管線」

```
df = fetch_datasets()['protein_homo']
X = PCA(n_components=2).fit_transform(df.data)
X = MinMaxScaler().fit_transform(X)
Y = df.target

X_train, X_test, Y_train, Y_test = train_test_split(
    X, Y, test_size=0.33, random_state=42, shuffle=True
)
print('X_Shape :', X.shape)
print('Y_Shape :', Y.shape)
print('Positive Ratio :', np.count_nonzero(Y == 1) / Y.shape[0])


lr = LogisticRegression().fit(X_train, Y_train)
Y_pred = lr.predict(X_test)

print('Report :', classification_report(Y_test, Y_pred))
print('ROC :', roc_auc_score(Y_test, Y_pred))

plot_base = np.linspace(0, 1, 1000)
value = (-lr.coef_[0][0] * plot_base - lr.intercept_) / lr.coef_[0][1]

sns.scatterplot(x= X_train[:, 0], y = X_train[:, 1], hue=Y_train)
plt.plot(plot_base, value)
plt.title('Positive Sample')
plt.ylim(0, 1)
plt.show()
```

```
X_Shape : (145751, 2)
Y_Shape : (145751,)
Positive Ratio : 0.008891877242694734
Report :               precision    recall  f1-score   support

          -1       0.99      1.00      1.00     47700
```

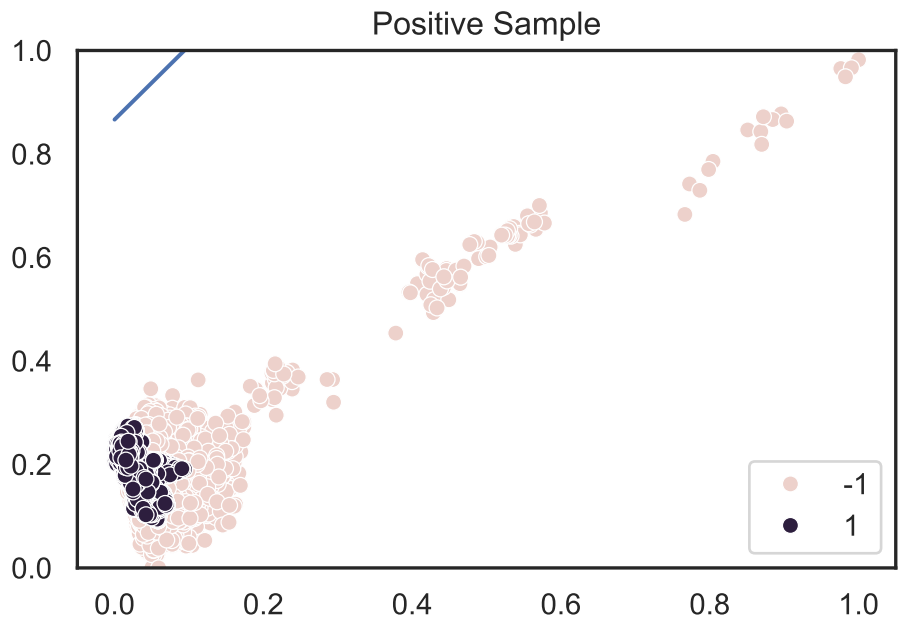
```
           1       0.00      0.00      0.00       398

    accuracy                           0.99     48098
   macro avg       0.50      0.50      0.50     48098
weighted avg       0.98      0.99      0.99     48098

ROC : 0.5

C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```




```python
df = fetch_datasets()['protein_homo']
X = PCA(n_components=2).fit_transform(df.data)
X = MinMaxScaler().fit_transform(X)
Y = df.target

X_train, X_test, Y_train, Y_test = train_test_split(
    X, Y, test_size=0.33, random_state=42, shuffle=True
)
# smote
X_res , Y_res = SMOTE(random_state=42).fit_resample(X_train,Y_train)
lr = LogisticRegression().fit(X_train, Y_train)
```

```
Y_pred = lr.predict(X_test)

print('Report :', classification_report(Y_test, Y_pred))
print('ROC :', roc_auc_score(Y_test, Y_pred))

plot_base = np.linspace(0, 1, 1000)
value = (-lr.coef_[0][0] * plot_base - lr.intercept_) / lr.coef_[0][1]

sns.scatterplot(x=X_res[:, 0],y= X_res[:, 1], hue=Y_res)
plt.plot(plot_base, value)
plt.title('Positive Sample')
plt.ylim(0, 1)
plt.show()
```

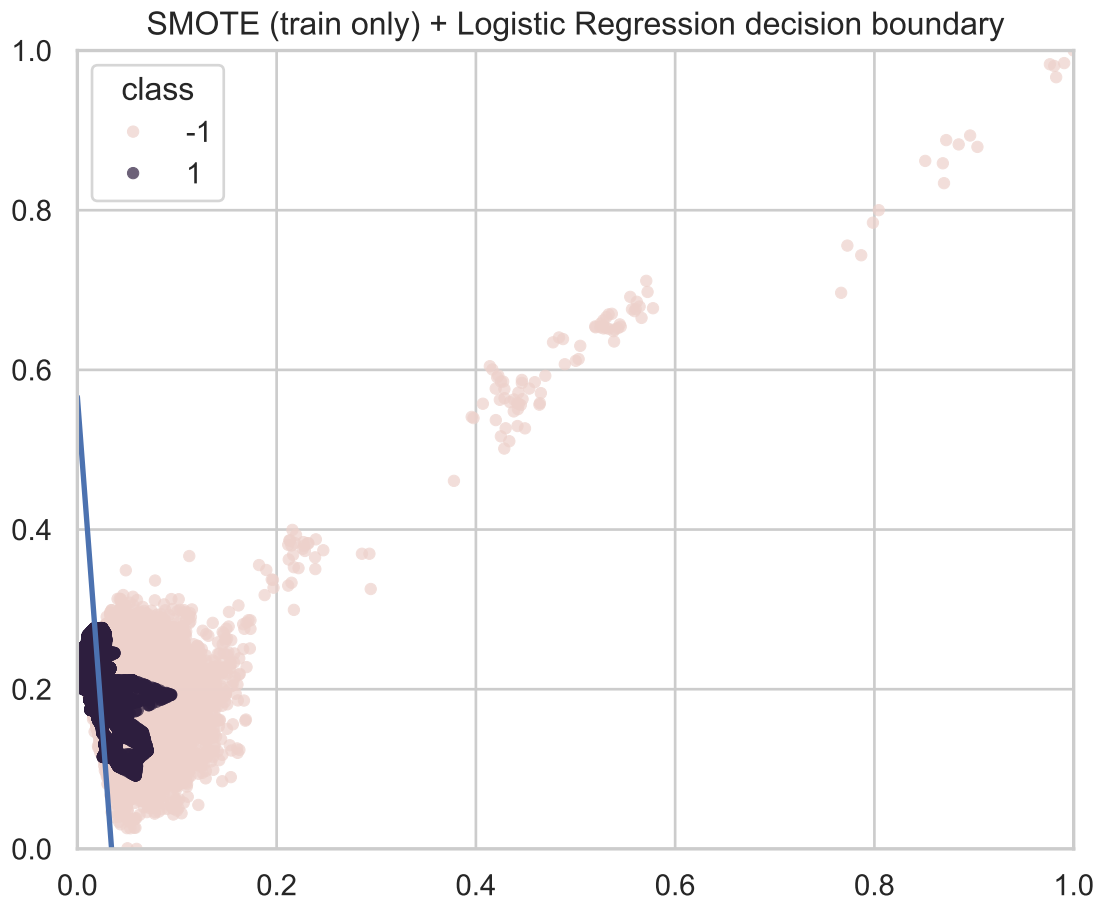
```
C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))

Report :               precision    recall  f1-score   support

          -1       0.99      1.00      1.00     47700
           1       0.00      0.00      0.00       398

    accuracy                           0.99     48098
   macro avg       0.50      0.50      0.50     48098
weighted avg       0.98      0.99      0.99     48098

ROC : 0.5
```




```
df = fetch_datasets()['protein_homo']
X0, y = df.data, df.target
X_tr0, X_te0, y_tr, y_te = train_test_split(
    X0, y, test_size=0.33, random_state=42, shuffle=True
)

# 2)     fit PCA/Scaler  transform   2D     [0,1]
pca = PCA(n_components=2)
scaler = MinMaxScaler()
X_tr = scaler.fit_transform(pca.fit_transform(X_tr0))
X_te = scaler.transform(pca.transform(X_te0))

# 3) SMOTE
X_res, y_res = SMOTE(random_state=42).fit_resample(X_tr, y_tr)

lr = LogisticRegression(max_iter=1000).fit(X_res, y_res)  # ←  X_res, y_res

# 4)       AUC
y_pred  = lr.predict(X_te)
y_proba = lr.predict_proba(X_te)[:, 1]
print('Report:\n', classification_report(y_te, y_pred))
print('ROC AUC:', roc_auc_score(y_te, y_proba))

# 5)     2D       +
sns.set(style="whitegrid")
plt.figure(figsize=(6, 5))
sns.scatterplot(x=X_res[:, 0], y=X_res[:, 1], hue=y_res, s=18, alpha=0.7, edgecolor=None
```

```
w = lr.coef_[0]; b = lr.intercept_[0]
x1 = np.linspace(0, 1, 500)
#   w[1]    0
eps = 1e-12 if abs(w[1]) < 1e-12 else 0.0
x2 = -(w[0] * x1 + b) / (w[1] + eps)

plt.plot(x1, x2, linewidth=2)
plt.title('SMOTE (train only) + Logistic Regression decision boundary')
plt.xlim(0, 1); plt.ylim(0, 1)
plt.legend(title='class')
plt.tight_layout()
plt.show()

pipe = Pipeline(steps=[
    ("pca", PCA(n_components=2)),
    ("scale", MinMaxScaler()),
    ("smote", SMOTE(random_state=42)),       #
    ("clf", LogisticRegression(max_iter=1000))
])

cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
scores = cross_val_score(pipe, X0, y, cv=cv, scoring="roc_auc")  #     fit
print("CV AUC:", scores.mean(), "+/-", scores.std())
```

```
Report:
              precision    recall  f1-score   support

          -1       1.00      0.64      0.78     47700
           1       0.02      0.81      0.04       398

    accuracy                           0.64     48098
   macro avg       0.51      0.73      0.41     48098
weighted avg       0.99      0.64      0.77     48098

ROC AUC: 0.813398228037462
```




```
CV AUC: 0.8348952067795306 +/- 0.006262153560792625
```

6 boardline smote

過採樣資料作羅吉斯回歸

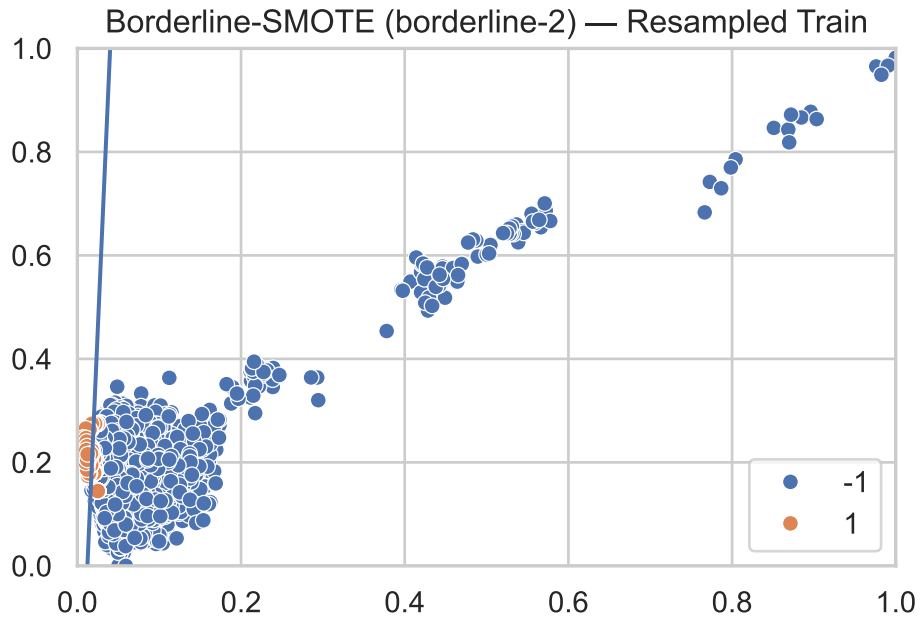
```
df = fetch_datasets()['protein_homo']
X = PCA(n_components=2).fit_transform(df.data)
X = MinMaxScaler().fit_transform(X)
Y = df.target

X_train, X_test, Y_train, Y_test = train_test_split(
    X, Y, test_size=0.33, random_state=42, shuffle=True
)

# --- Borderline-SMOTE (borderline-2) ---
```

```python
bl_smote = BorderlineSMOTE(random_state=42, kind='borderline-2')
X_res, Y_res = bl_smote.fit_resample(X_train, Y_train)

# --- Train on resampled data ---
lr = LogisticRegression(solver='liblinear', random_state=42).fit(X_res, Y_res)

# --- Predict & Metrics ---
y_prob = lr.predict_proba(X_test)[:, 1]
y_pred = (y_prob >= 0.5).astype(int)

print('Report:\n', classification_report(Y_test, y_pred))
print('ROC AUC:', roc_auc_score(Y_test, y_prob))

# --- Decision boundary (for visualization on [0,1]x[0,1]) ---
plot_base = np.linspace(0, 1, 1000)
# w0 * x + w1 * y + b = 0  →  y = -(w0*x + b)/w1
w0, w1 = lr.coef_[0]
b = lr.intercept_[0]
value = (-(w0 * plot_base) - b) / w1

# --- Plot ---
sns.scatterplot(x=X_res[:, 0], y=X_res[:, 1], hue=Y_res, palette='deep', s=35)
plt.plot(plot_base, value)
plt.title('Borderline-SMOTE (borderline-2) - Resampled Train')
plt.ylim(0, 1); plt.xlim(0, 1)
plt.show()
```

```
C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
Report:
               precision    recall  f1-score   support
```

```
          -1       0.00      0.00      0.00     47700
           0       0.00      0.00      0.00         0
           1       0.02      0.70      0.05       398

    accuracy                           0.01     48098
   macro avg       0.01      0.23      0.02     48098
weighted avg       0.00      0.01      0.00     48098

ROC AUC: 0.8044119970923801
```



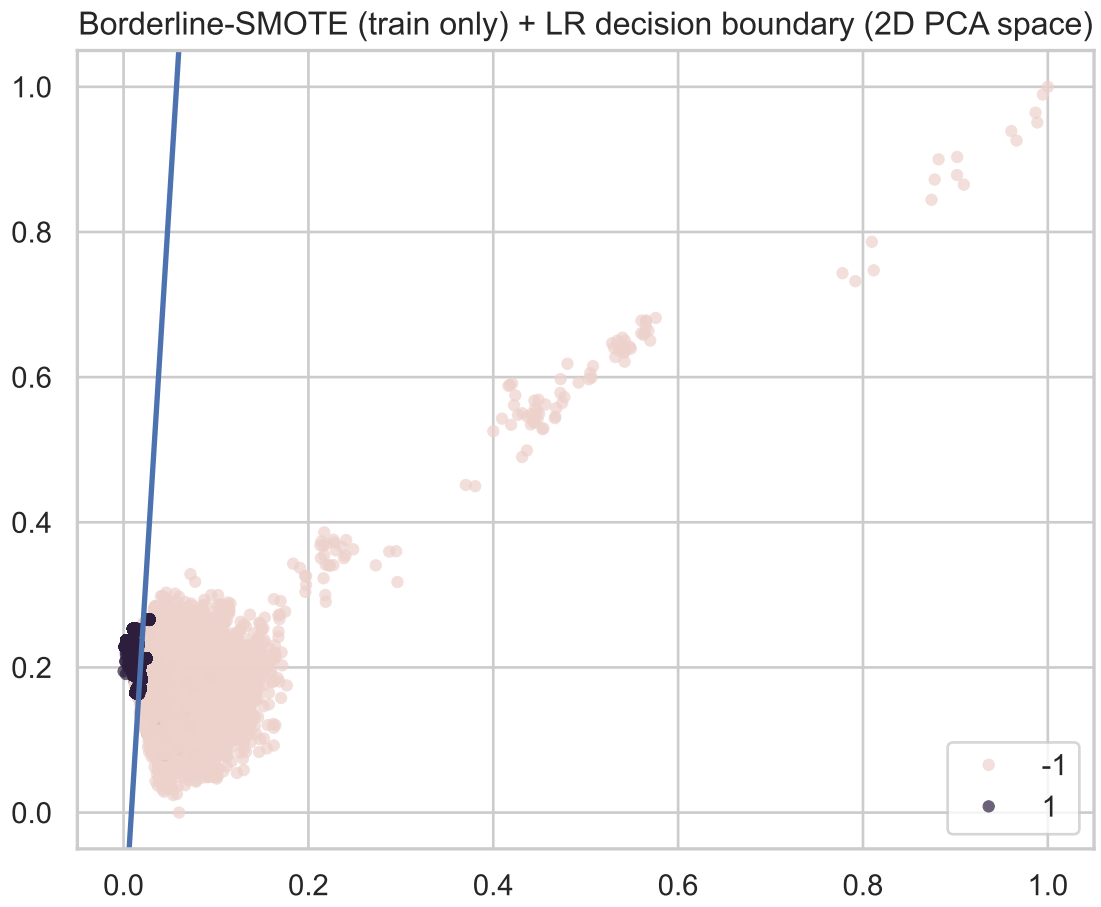

```python
RANDOM_STATE = 42

#
df = fetch_datasets()['protein_homo']
X0, y0 = df.data, df.target

# ------------------------------
#   A    (Hold-out)
# ------------------------------
X_tr, X_te, y_tr, y_te = train_test_split(
    X0, y0, test_size=0.33, stratify=y0, shuffle=True, random_state=RANDOM_STATE
)

#      PCA( / ) → Scaler → ( )SMOTE → LR
pipe_holdout = Pipeline(steps=[
    ("pca", PCA(n_components=2)),                #        2     /
```

```
    ("scale", MinMaxScaler()),
    ("smote", BorderlineSMOTE(random_state=RANDOM_STATE, kind="borderline-2")),
    ("clf", LogisticRegression(
        solver="liblinear", max_iter=1000, random_state=RANDOM_STATE
        # , class_weight="balanced"   #
    ))
])

#     fit PCA/Scaler/SMOTE        &
pipe_holdout.fit(X_tr, y_tr)

#              pca/scale
proba_te = pipe_holdout.predict_proba(X_te)[:, 1]
pred_te  = (proba_te >= 0.5).astype(int)

print("=== Hold-out   ===")
print(classification_report(y_te, pred_te))
print("ROC AUC:", roc_auc_score(y_te, proba_te))


# ------------------------------
#   B    (   )
# ------------------------------
#     CV         2D
pipe_cv = Pipeline(steps=[
    # ("pca", PCA(n_components=10)),              #         PCA
    ("scale", MinMaxScaler()),
    ("smote", SMOTE(random_state=RANDOM_STATE)),
    ("clf", LogisticRegression(
        solver="liblinear", max_iter=1000, random_state=RANDOM_STATE
    ))
])

cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=RANDOM_STATE)
scores = cross_validate(
    pipe_cv, X0, y0, cv=cv,
    scoring={"roc": "roc_auc", "pr": "average_precision"}, n_jobs=-1, return_train_score
)
print("\n=== 5-Fold CV    PCA    ===")
print("ROC AUC (mean ± sd):", scores["test_roc"].mean(), "±", scores["test_roc"].std())
print("PR  AUC (mean ± sd):", scores["test_pr"].mean(),  "±", scores["test_pr"].std())


# ------------------------------------------------
```

```
#   2D
# ------------------------------------------------
#           fit   PCA/Scaler
#     holdout    pca=2
print("\n=== 2D       ===")
#
pca_fitted    = pipe_holdout.named_steps["pca"]
scaler_fitted = pipe_holdout.named_steps["scale"]
clf_fitted    = pipe_holdout.named_steps["clf"]

#     X   fit   pca+scale    2D
X_tr_2d = scaler_fitted.transform(pca_fitted.transform(X_tr))

#   SMOTE           Pipeline     resampled
#              pca/scale
X_res_2d, y_res = BorderlineSMOTE(random_state=RANDOM_STATE, kind="borderline-2").fit_re

sns.set(style="whitegrid")
plt.figure(figsize=(6,5))
sns.scatterplot(x=X_res_2d[:,0], y=X_res_2d[:,1], hue=y_res, s=18, alpha=0.7, edgecolor=

#               w1*x + w2*y + b = 0 → y = -(w1*x + b)/w2
w = clf_fitted.coef_[0]; b = clf_fitted.intercept_[0]
x1 = np.linspace(X_res_2d[:,0].min()-0.05, X_res_2d[:,0].max()+0.05, 500)
eps = 1e-12 if abs(w[1]) < 1e-12 else 0.0
x2 = -(w[0]*x1 + b)/(w[1] + eps)
plt.plot(x1, x2, linewidth=2)

plt.xlim(X_res_2d[:,0].min()-0.05, X_res_2d[:,0].max()+0.05)
plt.ylim(X_res_2d[:,1].min()-0.05, X_res_2d[:,1].max()+0.05)
plt.title("Borderline-SMOTE (train only) + LR decision boundary (2D PCA space)")
plt.tight_layout()
plt.show()
```

```
=== Hold-out   ===
              precision    recall  f1-score   support

          -1       0.00      0.00      0.00     47670
           0       0.00      0.00      0.00         0
           1       0.03      0.75      0.05       428

    accuracy                           0.01     48098
   macro avg       0.01      0.25      0.02     48098
weighted avg       0.00      0.01      0.00     48098
```

```
ROC AUC: 0.8299555060197737

C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))


=== 5-Fold CV    PCA    ===
ROC AUC (mean ± sd): 0.9914037954296167 ± 0.0021338189909572854
PR  AUC (mean ± sd): 0.8563434416503577 ± 0.009340885431156966

=== 2D       ===
```



7 smote + tomeklinks

分別都在pca那裏的部分作先fit跟後fit

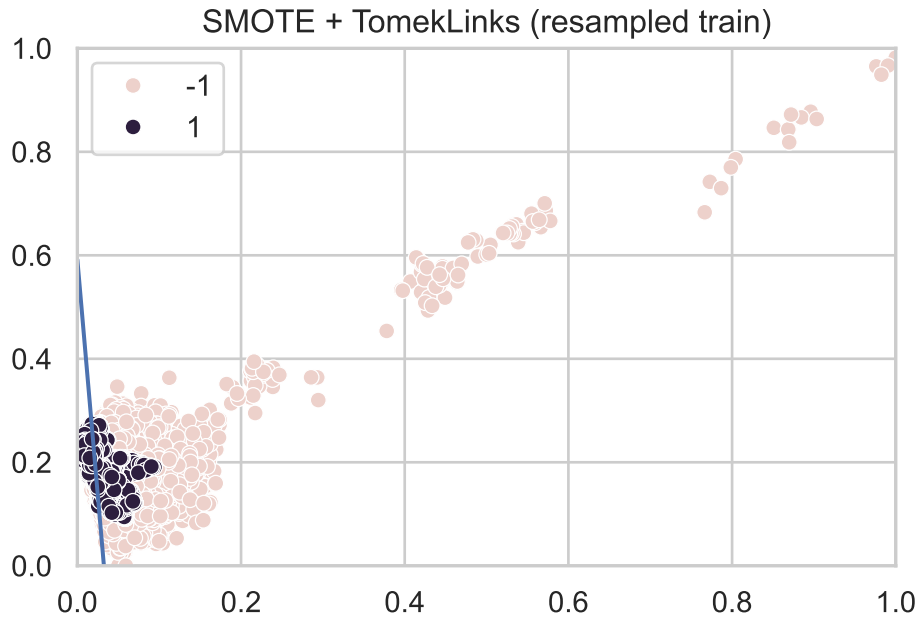
```
from imblearn.under_sampling import TomekLinks
# --- Data ---
df = fetch_datasets()['protein_homo']
X = PCA(n_components=2).fit_transform(df.data)
X = MinMaxScaler().fit_transform(X)
Y = df.target
X_train, X_test, Y_train, Y_test = train_test_split(
    X, Y, test_size=0.33, random_state=42, shuffle=True
)

# --- SMOTE + TomekLinks ---
# Pipeline: SMOTE+Tomek → LR
X_sm, y_sm = SMOTE(random_state=42).fit_resample(X_train, Y_train)
```

```
X_res, y_res = TomekLinks().fit_resample(X_sm, y_sm)

# --- Train ---
lr = LogisticRegression(solver='liblinear', random_state=42).fit(X_res, y_res)

# --- Metrics ---
y_prob = lr.predict_proba(X_test)[:, 1]
y_pred = (y_prob >= 0.5).astype(int)
print('Report:\n', classification_report(Y_test, y_pred))
print('ROC AUC:', roc_auc_score(Y_test, y_prob))

# --- Decision boundary ---
w0, w1 = lr.coef_[0]; b = lr.intercept_[0]
xx = np.linspace(0, 1, 1000)
yy = (-(w0 * xx) - b) / w1

# --- Plot ---
sns.scatterplot(x=X_res[:, 0], y=X_res[:, 1], hue=y_res, s=35)
plt.plot(xx, yy)
plt.title('SMOTE + TomekLinks (resampled train)')
plt.xlim(0, 1); plt.ylim(0, 1)
plt.show()
```

```
C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
Report:
               precision    recall  f1-score   support

          -1       0.00      0.00      0.00     47700
           0       0.00      0.00      0.00         0
           1       0.02      0.80      0.04       398
```

```
    accuracy                           0.01      48098
   macro avg       0.01      0.27      0.01      48098
weighted avg       0.00      0.01      0.00      48098

ROC AUC: 0.8130288233620935
```

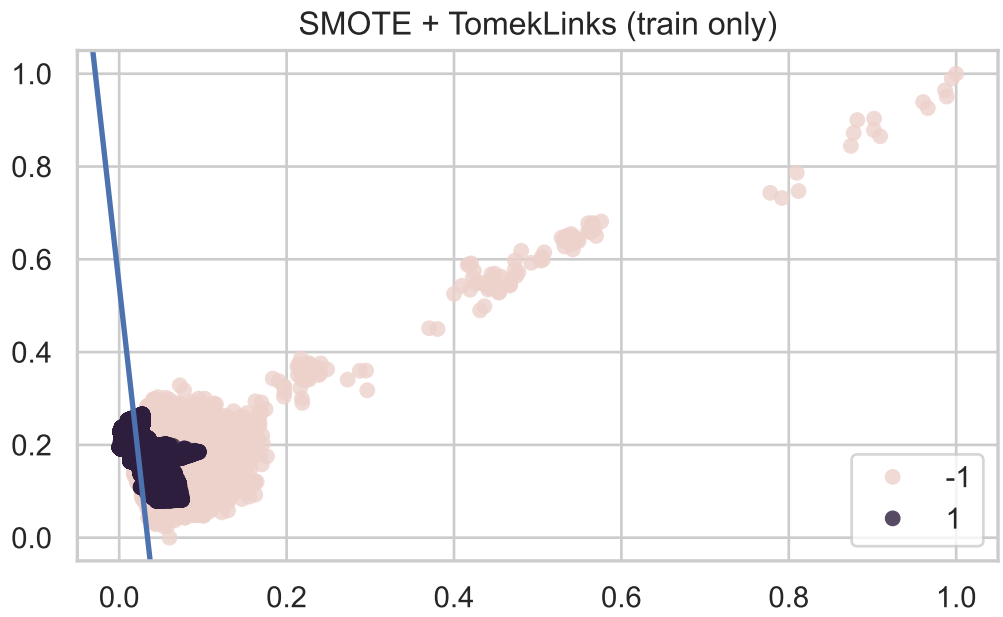



```python
from imblearn.under_sampling import TomekLinks
# --- Data ---
df = fetch_datasets()['protein_homo']
X0, y = df.data, df.target

#      fit
X_tr0, X_te0, y_tr, y_te = train_test_split(
    X0, y, test_size=0.33, random_state=42, shuffle=True, stratify=y
)

pca, scaler = PCA(n_components=2), MinMaxScaler()
X_tr = scaler.fit_transform(pca.fit_transform(X_tr0))   # train: fit
X_te = scaler.transform(pca.transform(X_te0))           # test: transform

# SMOTE → TomekLinks
X_sm, y_sm = SMOTE(random_state=42).fit_resample(X_tr, y_tr)
X_res, y_res = TomekLinks().fit_resample(X_sm, y_sm)

#
lr = LogisticRegression(solver='liblinear', max_iter=1000, random_state=42).fit(X_res, y
```

```python
y_prob = lr.predict_proba(X_te)[:, 1]
y_pred = (y_prob >= 0.5).astype(int)
print('Report:\n', classification_report(y_te, y_pred))
print('ROC AUC:', roc_auc_score(y_te, y_prob))

#
w0, w1 = lr.coef_[0]; b = lr.intercept_[0]
x1 = np.linspace(X_res[:,0].min()-0.05, X_res[:,0].max()+0.05, 600)
eps = 1e-12 if abs(w1) < 1e-12 else 0.0
y1 = (-(w0 * x1) - b) / (w1 + eps)

sns.scatterplot(x=X_res[:,0], y=X_res[:,1], hue=y_res, s=30, alpha=0.8, edgecolor=None)
plt.plot(x1, y1, linewidth=2)
plt.xlim(X_res[:,0].min()-0.05, X_res[:,0].max()+0.05)
plt.ylim(X_res[:,1].min()-0.05, X_res[:,1].max()+0.05)
plt.title('SMOTE + TomekLinks (train only)')
plt.tight_layout(); plt.show()
```

```
C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
Report:
               precision    recall  f1-score   support

          -1       0.00      0.00      0.00     47670
           0       0.00      0.00      0.00         0
           1       0.02      0.85      0.04       428

    accuracy                           0.01     48098
   macro avg       0.01      0.28      0.01     48098
weighted avg       0.00      0.01      0.00     48098

ROC AUC: 0.8429238495183985
```



8 smote + ENN

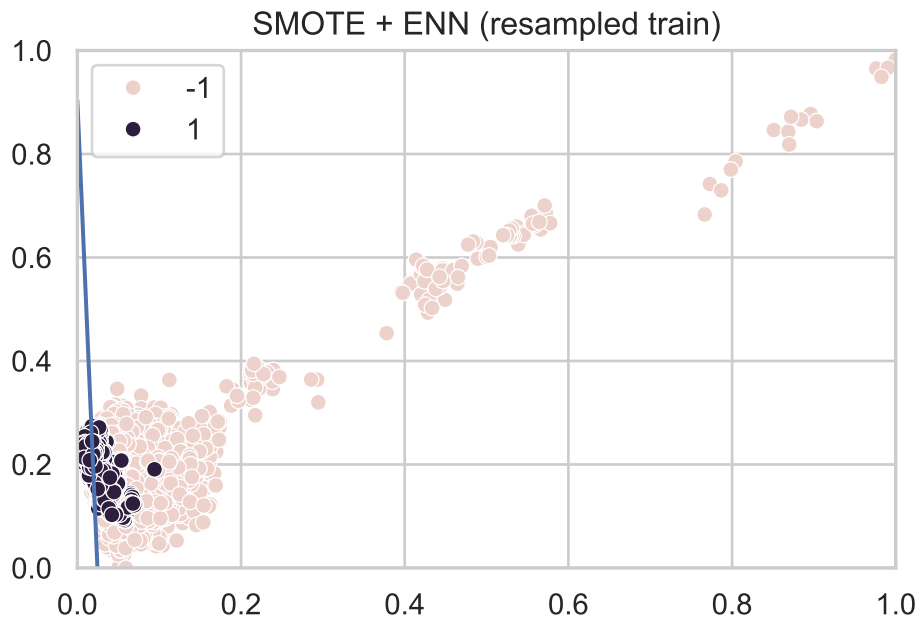
一樣的事情

```
from imblearn.under_sampling import EditedNearestNeighbours
# --- Data ---
df = fetch_datasets()['protein_homo']
X = PCA(n_components=2).fit_transform(df.data)
X = MinMaxScaler().fit_transform(X)
Y = df.target
X_train, X_test, Y_train, Y_test = train_test_split(
    X, Y, test_size=0.33, random_state=42, shuffle=True
)

# --- SMOTE + ENN ---
X_sm, y_sm = SMOTE(random_state=42).fit_resample(X_train, Y_train)
# n_neighbors   3
X_res, y_res = EditedNearestNeighbours(n_neighbors=3).fit_resample(X_sm, y_sm)

# --- Train ---
lr = LogisticRegression(solver='liblinear', random_state=42).fit(X_res, y_res)

# --- Metrics ---
y_prob = lr.predict_proba(X_test)[:, 1]
y_pred = (y_prob >= 0.5).astype(int)
```

```python
print('Report:\n', classification_report(Y_test, y_pred))
print('ROC AUC:', roc_auc_score(Y_test, y_prob))

# --- Decision boundary ---
w0, w1 = lr.coef_[0]; b = lr.intercept_[0]
xx = np.linspace(0, 1, 1000)
yy = (-(w0 * xx) - b) / w1

# --- Plot ---
sns.scatterplot(x=X_res[:, 0], y=X_res[:, 1], hue=y_res, s=35)
plt.plot(xx, yy)
plt.title('SMOTE + ENN (resampled train)')
plt.xlim(0, 1); plt.ylim(0, 1)
plt.show()
```

```
C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
Report:
               precision    recall  f1-score   support

          -1       0.00      0.00      0.00     47700
           0       0.00      0.00      0.00         0
           1       0.02      0.75      0.05       398

    accuracy                           0.01     48098
   macro avg       0.01      0.25      0.02     48098
weighted avg       0.00      0.01      0.00     48098

ROC AUC: 0.8106848708953572
```

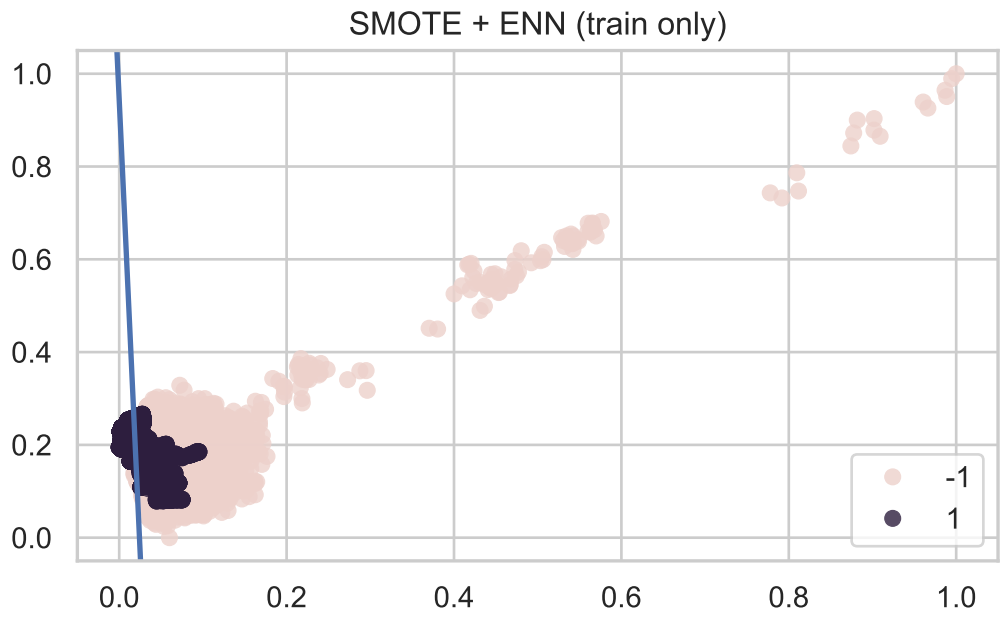



```
from imblearn.under_sampling import EditedNearestNeighbours
# --- Data ---
df = fetch_datasets()['protein_homo']
X0, y = df.data, df.target

# 1)    fit
X_tr0, X_te0, y_tr, y_te = train_test_split(
    X0, y, test_size=0.33, random_state=42, shuffle=True, stratify=y
)

pca, scaler = PCA(n_components=2), MinMaxScaler()
X_tr = scaler.fit_transform(pca.fit_transform(X_tr0))   #   train fit
X_te = scaler.transform(pca.transform(X_te0))           # test   transform

# 2) SMOTE → ENN
X_sm, y_sm = SMOTE(random_state=42).fit_resample(X_tr, y_tr)
#     EditedNearestNeighbours(n_neighbors=3, sampling_strategy='majority')
X_res, y_res = EditedNearestNeighbours(n_neighbors=3).fit_resample(X_sm, y_sm)

# 3)
lr = LogisticRegression(solver='liblinear', max_iter=1000, random_state=42).fit(X_res, y
y_prob = lr.predict_proba(X_te)[:, 1]
y_pred = (y_prob >= 0.5).astype(int)
print('Report:\n', classification_report(y_te, y_pred))
print('ROC AUC:', roc_auc_score(y_te, y_prob))

# 4)
```

```python
w0, w1 = lr.coef_[0]; b = lr.intercept_[0]
x1 = np.linspace(X_res[:,0].min()-0.05, X_res[:,0].max()+0.05, 600)
eps = 1e-12 if abs(w1) < 1e-12 else 0.0
y1 = (-(w0*x1) - b) / (w1 + eps)

sns.scatterplot(x=X_res[:,0], y=X_res[:,1], hue=y_res, s=35, alpha=0.8, edgecolor=None)
plt.plot(x1, y1, linewidth=2)
plt.xlim(X_res[:,0].min()-0.05, X_res[:,0].max()+0.05)
plt.ylim(X_res[:,1].min()-0.05, X_res[:,1].max()+0.05)
plt.title('SMOTE + ENN (train only)'); plt.tight_layout(); plt.show()
```

```
C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\jerry\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
Report:
               precision    recall  f1-score   support

          -1       0.00      0.00      0.00     47670
           0       0.00      0.00      0.00         0
           1       0.03      0.79      0.05       428

    accuracy                           0.01     48098
   macro avg       0.01      0.26      0.02     48098
weighted avg       0.00      0.01      0.00     48098

ROC AUC: 0.8394614258070967
```



9 crossfit

這邊先做資料與評估設定，三條 Pipeline，交叉驗證與彙整

```
from sklearn.model_selection import cross_val_score, StratifiedKFold
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import make_classification
from imblearn.pipeline import Pipeline
from imblearn.over_sampling import BorderlineSMOTE
from sklearn.model_selection import StratifiedKFold, cross_validate
from IPython.display import display, Markdown


df = fetch_datasets()['protein_homo']
X, y = df.data, df.target

# =====    =====
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
scoring = {
    'roc_auc': 'roc_auc',
    'f1_macro': 'f1_macro',
    'pr_auc': 'average_precision',  # Precision-Recall AUC
}
clf = LogisticRegression(solver='liblinear', random_state=42)
```

```
# =====   Pipeline =====
pipelines = {
    # Baseline
    'baseline': Pipeline([
        ('scale', MinMaxScaler()),
        ('pca', PCA(n_components=2, random_state=42)),   #     2D
        ('clf', clf),
    ]),
    # SMOTE + TomekLinks
    'smote_tomek': Pipeline([
        ('scale', MinMaxScaler()),
        ('pca', PCA(n_components=2, random_state=42)),
        ('resample', SMOTETomek(random_state=42)),
        ('clf', clf),
    ]),
    # SMOTE + ENN
    'smote_enn': Pipeline([
        ('scale', MinMaxScaler()),
        ('pca', PCA(n_components=2, random_state=42)),
        ('resample', SMOTEENN(random_state=42)),
        ('clf', clf),
    ]),
}

# =====       =====
rows = []
for name, pipe in pipelines.items():
    cv_res = cross_validate(pipe, X, y, cv=cv, scoring=scoring, n_jobs=-1, return_train_
    rows.append({
        'method': name,
        'roc_auc_mean': np.mean(cv_res['test_roc_auc']),
        'roc_auc_std':  np.std(cv_res['test_roc_auc']),
        'f1_macro_mean': np.mean(cv_res['test_f1_macro']),
        'f1_macro_std':  np.std(cv_res['test_f1_macro']),
        'pr_auc_mean': np.mean(cv_res['test_pr_auc']),
        'pr_auc_std':  np.std(cv_res['test_pr_auc']),
    })



result = pd.DataFrame(rows).sort_values('roc_auc_mean', ascending=False)
print(result.to_string(index=False, float_format=lambda x: f"{x:.4f}"))
```

```
     method  roc_auc_mean  roc_auc_std  f1_macro_mean  f1_macro_std  pr_auc_mean  pr_auc
```

baseline	0.8077	0.0131	0.4993	0.0019	0.0772	0.
smote_enn	0.8076	0.0131	0.4453	0.0012	0.0772	0.
smote_tomek	0.8076	0.0131	0.4525	0.0010	0.0771	0.