Midterm Project Report

學號：B12501001　系級：土木三　姓名：蔣尚誠

1. (1%) 請說明你如何進行資料前處理。

i. 讀檔與欄位檢查

- 從 train.xlsx、test.xlsx 載入資料。
- 檢查標籤欄｛崩塌｝是否存在，避免後續報錯。
- 將特徵與標籤分離：X_full_all = train.drop('崩塌')、y = train['崩塌']。

ii. 分層切分（避免類別比例偏差）

- 使用 train_test_split(..., stratify=y, test_size=0.2, random_state=42)，確保訓練/驗證集的正負樣本比例與原始資料一致，減少評估偏差。

iii. 缺失值補齊（以訓練集統計量為準，防資料洩漏）

- 建立單一 SimpleImputer(strategy='median')，只在訓練集 fit，再 transform 到驗證集與測試集：

```
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer

X_train, X_val, y_train, y_val = train_test_split(
    X_full_all, y, stratify=y, test_size=0.2, random_state=42
)

imputer = SimpleImputer(strategy='median')
X_train_imp = pd.DataFrame(imputer.fit_transform(X_train), columns=X_train.columns, index=X_train.index)
X_val_imp   = pd.DataFrame(imputer.transform(X_val),     columns=X_val.columns,   index=X_val.index)
X_test_imp  = pd.DataFrame(imputer.transform(test[X_full_all.columns]), columns=X_full_all.columns, index=test.index)
```

- 選擇「中位數補值」以降低離群值影響。

2. (1%) 請說明你所使用的 model 以及 hyper-parameter tuning 的心得。

i. RandomForestClassifier

我用它來在「已補值的訓練集」上估特徵重要度，篩出 Top-N 特徵（取前 10 或 15）。會選它是因為 RF 對特徵尺度不敏感、能處理非線性與交互作用、對缺少嚴格的前處理比較寬容，很適合作為「第一層特徵過濾器」。

ii. XGBoost

我用它來在最終的分類器。先用「單一模型」做驗證，再做 5-fold 集成做出最後的測試預測。它能處理高維非線性、對缺失值與不平衡情況提供原生支持（ scale_pos_weight ）。

iii. CatBoost

我還有另一個程式是跑 CatBoost，先用一個小型的 CatBoostClassifier（搭配 Pool 指定欄名）在已補值的訓練資料上快速擬合，透過 get_feature_importance 產生特徵重要度排序，與你預先指定的必選欄位一起組成最後要用的特徵子集；接著在主流程裡，把 CatBoost 當成最終分類器放進含重採樣的管線（先用自訂的 SafeSMOTEENN 處理不平衡、再用 FunctionTransformer 做 NaN/Inf 防護、最後交給 clf=CatBoostClassifier），並以 RandomizedSearchCV 對迭代次數、深度、學習率、子特徵抽樣比例等超參數做隨機搜尋，折內評估指標設定為 PR-AUC，同時也回報 ROC-AUC，但以 PR-AUC 最大者作為 best_estimator_，也就是最終要用來預測的 CatBoost 模型。

iv. hyper-parameter tuning

我採「先篩特徵 → 再調模型 → 最後定門檻」的三段式流程。首先，用 RandomForest 在只用訓練集統計量補值後估特徵重要度，RandomForest 在訓練集（補值後）估重要度並做分層 CV 比較 Top-K。實測 Top-10 與 Top-15 的表現差異不大，為了簡化模型以及避免選到噪音特徵，我採用 Top-10 作為增補集合。XGBoost 的調參我有關注 ROC-AUC 與 PR-AUC 的平衡，過程中同時監看兩者以求折衷，但因 ROC-AUC 長期維持高檔、變化度有限，我將焦點放在提升 PR-AUC 的品質。並且進行 n_estimators、max_depth、learning_rate、subsample 微調，並聯動 min_child_weight、colsample_bytree、reg_alpha/λ 與 scale_pos_weight 抑制過擬合、提升召回精準的平衡。經驗結論是：

```
def make_xgb_params(spw=None):
    return dict(
        n_estimators=10000,
        max_depth=8,
        learning_rate=0.05,
        eval_metric="logloss",
        random_state=42,
        subsample=0.9,
        colsample_bytree=0.9,
        min_child_weight=1,
        reg_lambda=1.0,
        reg_alpha=0.0,
        scale_pos_weight=scale_pos_weight if spw is None else spw,
        tree_method="hist",   # CPU
        n_jobs=-1
    )
```

最後，threshold 的調參我原先有跑一個最佳化 threshold，跑出來去預測 test 的結果發現 1 的權重太重，test 中的 1 超過 600 個，但我們知道 test 中的數據為 1:1 的均衡資料，所以後來我直接手動修改 threshold，讓他跑出來的 test 中的 1 接近 500 個。

3. (1%) 請寫出上課沒有提及但是你有使用於本次競賽的技巧。

我認為最多的部分是資料前處理的部分，要看要如何補值以及在何時補值較為適當，切分資料時用何種方法才可以提高預測的準確率。例如：補值只在訓練資料 fit 補值器（用中位數），再 transform 驗證與測試。切分時機為先切分再補值，在特徵選擇與建模前完成補值，並於每折管線內重複同一流程，確保評估一致。然後 XGBoost 的各項參數對預測的影響為何，並且過擬合的問題，不能一昧地追求高準確度，需要知道哪些參數會造成問題，哪些可以降低擬合問題。