Experiment Objective

• membandingkan metode CV

Hasil dan Analisa :

• Saya salah hitung, sudah benar 243 kandidat yang dicek karena banyak kandidat = 3^5=243
• Gunakan parameter n_job=-1. Ini akan otomatis membagi “job” secara merata ke seluruh thread pada CPU
• Hasil nya berbeda jika menggunakan K-fold dan stratifiedKFold
• Ketika input integer pada cv maka otomatis akan menggunakan stratifiedCV
• Untuk imbalance dataset, stratifiedKFold lebih tepat digunakan dibandingkan KFold. read more : https://machinelearningmastery.com/cross-validation-for-imbalanced-classification/ (https://machinelearningmastery.com/cross-validation-for-imbalanced-classification/)
• Pakai 10 fold hasilnya lebih baik. tp pasti lebih lama waktu eksekusinya

Code :

In [1]:
```
import pandas as pd
import numpy as np
import pickle
from xgboost import XGBClassifier
from sklearn.model_selection import KFold
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RandomizedSearchCV
```

In [3]:
```
X_bank=pd.read_csv('dataset/X_bank_preprocessed.csv').to_numpy()
y_bank=pd.read_csv('dataset/y_bank_preprocessed.csv').to_numpy().ravel()
```

ravel() merubah array dari (n,1) menjadi (n,)

In [4]:
```
model=XGBClassifier()
```

default hyperparameter XGBoost

In [36]:
```
model.fit(X_bank, y_bank, eval_metric='auc', verbose=True)
pickle.dump(model, open("xgb_temp.pkl", "wb"))
clf2 = pickle.load(open("xgb_temp.pkl", "rb"))
assert np.allclose(model.predict(X_bank), clf2.predict(X_bank))
print(clf2.get_xgb_params())
```

```
{'objective': 'binary:logistic', 'base_score': 0.5, 'booster': 'gbtree', 'col
sample_bylevel': 1, 'colsample_bynode': 1, 'colsample_bytree': 1, 'gamma': 0,
'gpu_id': -1, 'interaction_constraints': '', 'learning_rate': 0.300000012, 'm
ax_delta_step': 0, 'max_depth': 6, 'min_child_weight': 1, 'monotone_constrain
ts': '()', 'n_jobs': 0, 'num_parallel_tree': 1, 'random_state': 0, 'reg_alph
a': 0, 'reg_lambda': 1, 'scale_pos_weight': 1, 'subsample': 1, 'tree_method':
'exact', 'validate_parameters': 1, 'verbosity': None}
```

KFold

In [14]:
```
kfold=KFold(n_splits=5)
results=cross_val_score(model,X_bank,y_bank,cv=kfold,scoring="roc_auc")
print(f"AUC: {round(results.mean(),4)}, std: {round(results.std(),4)}")
```

```
AUC: 0.8562, std: 0.0558
```

StratifiedKFold

In [7]:
```
skf=StratifiedKFold(n_splits=5)
results=cross_val_score(model,X_bank,y_bank,cv=skf,scoring="roc_auc")
print(f"AUC: {round(results.mean(),4)}, std: {round(results.std(),4)}")
```

```
AUC: 0.4837, std: 0.1006
```

CV with integer input (5 fold vs 10 fold)

In [6]:
```
results=cross_val_score(model,X_bank,y_bank,cv=5,scoring="roc_auc")
print(f"AUC: {round(results.mean(),4)}, std: {round(results.std(),4)}")
```

```
AUC: 0.4837, std: 0.1006
```

In [8]:
```
results=cross_val_score(model,X_bank,y_bank,cv=10,scoring="roc_auc")
print(f"AUC: {round(results.mean(),4)}, std: {round(results.std(),4)}")
```

```
AUC: 0.5397, std: 0.1814
```

In [ ]:
```

```