1 라이브러리 로딩

In [1]:

```
import numpy as np # Numpy
import pandas as pd # Pandas
import matplotlib as mpl #Matplotlib 세팅용
import matplotlib.pyplot as plt # 시각화 도구
import seaborn as sns # 시각화 도구
from sklearn.model_selection import train_test_split # 데이터셋 분리
from sklearn.model_selection import KFold # KFold 교차검증
from sklearn.cluster import KMeans # 클러스터링
from sklearn.metrics import silhouette_score # 실루엣 점수
import xgboost as xgb # XGBoost
from sklearn.model_selection import GridSearchCV # 그리드 서치
from sklearn.metrics import accuracy_score, precision_score # 평가 지표
from sklearn.metrics import recall_score, confusion_matrix, roc_auc_score, f1_
from imblearn.combine import SMOTEENN, SMOTETomek # 복합샘플링
from hyperopt import hp, fmin, tpe, Trials # HyperOPT

import warnings # 경고문 제거용


%matplotlib inline
%config Inlinebackend.figure_format = 'retina'

# 한글 폰트 설정
mpl.rc('font', family='D2Coding')
# 유니코드에서 음수 부호 설정
mpl.rc('axes', unicode_minus = False)

warnings.filterwarnings('ignore')
sns.set(font="D2Coding", rc={"axes.unicode_minus":False}, style='darkgrid')
plt.rc('figure', figsize=(10,8))
```

executed in 4.26s, finished 16:49:50 2022-11-24

2 데이터 불러오기

In [2]:

```
data = pd.read_excel('train_test_na_filled.xlsx', sheet_name='Train')
```

executed in 2.39s, finished 16:49:53 2022-11-24

3 전처리

3.1 필요없는 feature 제거

In [ ]:

```
data.drop(['PassengerId', 'Cabin', 'Combi', 'Name',], axis=1, inplace=True)
```

3.2 결측값 제거

In [ ]:

```
# 결측값들 제거(Cabin)
data.dropna(axis=0, inplace=True)
```

3.3 원핫 인코딩

In [9]:

```
data['Cabin3'].replace({'P': True,'S': False}, inplace=True)
data['Cabin3'] = data['Cabin3'].astype(bool)
```

executed in 14ms, finished 16:49:53 2022-11-24

In [10]:

```
train_encoding = pd.get_dummies(data['HomePlanet'])
data=data.drop('HomePlanet',axis=1)
data = data.join(train_encoding)

train_encoding = pd.get_dummies(data['Destination'])
data=data.drop('Destination',axis=1)
data = data.join(train_encoding)

train_encoding = pd.get_dummies(data['Cabin1'])
data=data.drop('Cabin1',axis=1)
data = data.join(train_encoding)
```

executed in 27ms, finished 16:49:53 2022-11-24

3.4 원핫 인코딩

In [12]:

```
col = ['Cabin2', 'Age', 'RoomService','FoodCourt','ShoppingMall','Spa','VRDeck
def data_scaled(df, col):
    for i in col:
        data_mean = df[i].mean()
        data_std = df[i].std()
        scaled = (df[i]-data_mean)/data_std
        df[i]=scaled
    return df
```

executed in 14ms, finished 16:49:53 2022-11-24

In [13]:

```
1 data_scaled(data, col)
```

executed in 44ms, finished 16:49:53 2022-11-24

Out[13]:

	CryoSleep	Cabin2	Cabin3	Age	VIP	RoomService	FoodCourt	Shoppi
0	False	-1.170228	True	0.712274	False	-0.333743	-0.280785	
1	False	-1.170228	False	-0.332624	False	-0.168530	-0.275148	
2	False	-1.170228	False	2.035811	True	-0.268567	1.959032	
3	False	-1.170228	False	0.294315	False	-0.333743	0.522818	
4	False	-1.168274	False	-0.889902	False	0.125518	-0.236941	
...	...	...	...	...	...	...	...	
8688	False	-0.978724	True	0.851594	True	-0.333743	3.990274	
8689	True	1.758999	False	-0.750583	False	-0.333743	-0.280785	
8690	False	1.760953	False	-0.193304	False	-0.333743	-0.280785	
8691	False	0.017878	False	0.224655	False	-0.333743	0.376253	
8692	False	0.017878	False	1.060573	False	-0.142763	2.655529	

8590 rows × 25 columns

In [15]:

```
1 data.columns
```

executed in 14ms, finished 16:49:53 2022-11-24

Out[15]:

```
Index(['CryoSleep', 'Cabin2', 'Cabin3', 'Age', 'VIP', 'RoomService',
       'FoodCourt', 'ShoppingMall', 'Spa', 'VRDeck', 'Transported', 'E
arth',
       'Europa', 'Mars', '55 Cancri e', 'PSO J318.5-22', 'TRAPPIST-1
e', 'A',
       'B', 'C', 'D', 'E', 'F', 'G', 'T'],
      dtype='object')
```

In [16]:

```
1 df = data.iloc[:,:]
```

executed in 14ms, finished 16:49:53 2022-11-24

In [17]:

```
1 df['Transported'] = df.Transported.replace({True:1, False:0})
```

executed in 14ms, finished 16:49:53 2022-11-24

4 데이터셋 분리

In [24]:

```
X_train, X_test, y_train, y_test = train_test_split(df.drop(['Transported'], c
                                                    df.Transported, random_sta
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train)
```

executed in 13ms, finished 16:49:54 2022-11-24

In [25]:

```
X_train.shape
```

executed in 14ms, finished 16:49:54 2022-11-24

Out[25]:

```
(4831, 24)
```

In [26]:

```
y_train.shape
```

executed in 14ms, finished 16:49:54 2022-11-24

Out[26]:

```
(4831,)
```

5 XGBoost

5.1 HyperOPT

In [28]:

```
xgb_search_space = {'max_depth': hp.quniform('max_depth', 5, 15, 1),
                    'min_child_weight': hp.quniform('min_child_weight', 1, 20,
                    'colsample_bytree': hp.uniform('colsample_bytree', 0.5, 0.9
                    'learning_rate': hp.uniform('learning_rate', 0.01, 0.4),
                    'gamma': hp.uniform('gamma', 0, 4)}
```

executed in 10ms, finished 16:49:54 2022-11-24

In [29]:

```
# fmin()에서 호출 시 search_space 값으로 XGBClassifier 교차 검증 학습 후 -1 * roc_c
def bin_objective_func(search_space):
    xgb_clf = xgb.XGBClassifier(n_estimators=100,max_depth=int(search_space['m
                                min_child_weight=int(search_space['min_child_weigh
                                colsample_bytree=search_space['colsample_bytree'],
                                learning_rate=search_space['learning_rate'],
                                gamma=search_space['gamma'])

    # 3개 k-fold 방식으로 평가된 roc_auc 지표를 담는 list
    roc_auc_list = []

    # 3개 k-fold 방식 적용
    kf = KFold(n_splits=3)

    # X_train을 다시 학습과 검증용 데이터로 분리
    for tr_index, val_index in kf.split(X_train):
        # kf.split(X_train)으로 추출된 학습과 검증 index 값으로 학습과 검증 데이터 세
        X_tr, y_tr = X_train.iloc[tr_index], y_train.iloc[tr_index]
        X_val, y_val = X_train.iloc[val_index], y_train.iloc[val_index]

        # early stopping은 30회로 설정하고 추출된 학습과 검증 데이터로 XGBClassifier
        xgb_clf.fit(X_tr, y_tr, early_stopping_rounds=30, eval_metric="auc",
                    eval_set=[(X_tr, y_tr), (X_val, y_val)])

        # 1로 예측한 확률값 추출 후 roc auc 계산하고 평균 roc auc 계산을 위해 list에
        score = roc_auc_score(y_val, xgb_clf.predict_proba(X_val)[:,1])
        roc_auc_list.append(score)

    # 3개 k-fold로 계산된 roc_auc 값의 평균값을 반환하되,
    # HyperOPT는 목적함수의 최솟값을 위한 입력값을 찾으므로 -1을 곱한 뒤 반환
    return -1*np.mean(roc_auc_list)
```

executed in 14ms, finished 16:49:54 2022-11-24

In [30]:

```
trials = Trials()

# fmin() 함수를 호출. max_evals 지정된 횟수만큼 반복 후 목적함수의 최솟값을 가지는 최적
best = fmin(fn=bin_objective_func,
            space=xgb_search_space,
            algo=tpe.suggest,
            max_evals=50, # 최대 반복 횟수를 지정합니다
            trials=trials, rstate=np.random.default_rng(seed=109))

print('best:', best)
```

executed in 14ms, finished 16:49:54 2022-11-24

5.2 모델 학습

In [31]:

```
xgbo = xgb.XGBClassifier(colsample_bytree=0.708890925020427, gamma=2.039476183
                         learning_rate=0.16202345555142844,
                         max_depth=10, min_child_weight=6, n_estimators=100,
                         random_state=109)
xgbo.fit(X_train, y_train)
```

executed in 329ms, finished 16:49:55 2022-11-24

Out[31]:

```
XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1,
              colsample_bytree=0.708890925020427, early_stopping_round
s=None,
              enable_categorical=False, eval_metric=None, feature_type
s=None,
              gamma=2.039476183481736, gpu_id=-1, grow_policy='depthwi
se',
              importance_type=None, interaction_constraints='',
              learning_rate=0.16202345555142844, max_bin=256,
              max_cat_threshold=64, max_cat_to_onehot=4, max_delta_ste
p=0,
              max_depth=10, max_leaves=0, min_child_weight=6, missing=
nan,
              monotone_constraints='()', n_estimators=100, n_jobs=0,
              num_parallel_tree=1, predictor='auto', random_state=109,
...)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

5.3 평가 지표

In [32]:

```
# 평가용 함수
def  get_clf_eval(y_test, pred=None, pred_proba=None):
    confusion = confusion_matrix(y_test, pred)
    accuracy = accuracy_score(y_test, pred)
    precision = precision_score(y_test, pred)
    recall = recall_score(y_test, pred)
    f1 = f1_score(y_test, pred)
#     roc_auc = roc_auc_score(y_test, pred_proba)

    print('오차 행렬')
    print(confusion)

    print('정확도: {0:.4f}, 정밀도: {1:.4f}, \
    재현율: {2:.4f}, F1: {3:.4f}'.format(accuracy, precision, recall, f1))
```

executed in 14ms, finished 16:49:55 2022-11-24

In [33]:

```
train_pred = xgbo.predict(X_train)
train_proba = xgbo.predict_proba(X_train)

test_pred = xgbo.predict(X_test)
test_proba = xgbo.predict_proba(X_test)

val_pred = xgbo.predict(X_val)
val_proba = xgbo.predict_proba(X_val)
```

executed in 60ms, finished 16:49:55 2022-11-24

5.3.1 훈련셋

In [34]:

```
get_clf_eval(y_train, train_pred, train_proba)
```

executed in 14ms, finished 16:49:55 2022-11-24

```
오차 행렬
[[2109  258]
 [ 252 2212]]
정확도: 0.8944, 정밀도: 0.8955,     재현율: 0.8977, F1: 0.8966
```

5.3.2 테스트셋

In [35]:

```
get_clf_eval(y_test, test_pred, test_proba)
```

executed in 14ms, finished 16:49:55 2022-11-24

```
오차 행렬
[[849 218]
 [218 863]]
정확도: 0.7970, 정밀도: 0.7983,     재현율: 0.7983, F1: 0.7983
```

5.3.3 검증셋

In [36]:

```
1 get_clf_eval(y_val, val_pred, val_proba)
```

executed in 14ms, finished 16:49:55 2022-11-24

```
오차 행렬
[[667 156]
 [155 633]]
정확도: 0.8070, 정밀도: 0.8023,     재현율: 0.8033, F1: 0.8028
```

5.3.4 feature 중요도

In [48]:

```
1 xgbo.feature_importances_.shape
```

executed in 7ms, finished 17:21:55 2022-11-24

Out[48]:

```
(24,)
```

In [51]:

```
1 fi = pd.DataFrame(xgbo.feature_importances_, index=X_train.columns)
```

executed in 7ms, finished 17:26:41 2022-11-24

In [52]:

```
1 fi
```

executed in 12ms, finished 17:26:42 2022-11-24

Out[52]:

	0
CryoSleep	0.404858
Cabin2	0.013693
Cabin3	0.024119
Age	0.013701
VIP	0.002429
RoomService	0.043292
FoodCourt	0.032967
ShoppingMall	0.022653
Spa	0.040408
VRDeck	0.031387
Earth	0.097255
Europa	0.057619
Mars	0.018460
55 Cancri e	0.010320
PSO J318.5-22	0.010474
TRAPPIST-1e	0.016983
A	0.016070
B	0.015431
C	0.021391
D	0.004966
E	0.037450
F	0.045982
G	0.018092
T	0.000000