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 3.58s, finished 10:19:44 2022-11-23

2 데이터 불러오기

In [2]:

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

executed in 1.61s, finished 10:19:45 2022-11-23

3 전처리

In [3]:

```
1 data.info()
```

executed in 15ms, finished 10:19:45 2022-11-23

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8693 entries, 0 to 8692
Data columns (total 18 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   PassengerId   8693 non-null   object
 1   HomePlanet    8693 non-null   object
 2   CryoSleep     8693 non-null   bool
 3   Cabin1        8590 non-null   object
 4   Cabin2        8590 non-null   float64
 5   Combi         8590 non-null   object
 6   Cabin3        8590 non-null   object
 7   Cabin         8590 non-null   object
 8   Destination   8693 non-null   object
 9   Age           8693 non-null   int64
 10  VIP           8693 non-null   bool
 11  RoomService   8693 non-null   int64
 12  FoodCourt     8693 non-null   int64
 13  ShoppingMall  8693 non-null   int64
 14  Spa           8693 non-null   int64
 15  VRDeck        8693 non-null   int64
 16  Name          8493 non-null   object
 17  Transported   8693 non-null   bool
dtypes: bool(3), float64(1), int64(6), object(8)
memory usage: 1.0+ MB
```

3.1 필요없는 features 제거

In [4]:

```
1 # 필요없는 features 제거
2 data.drop(['PassengerId', 'Cabin', 'Cabin1', 'Cabin2', 'Name',], axis=1, inplc
```

executed in 15ms, finished 10:19:45 2022-11-23

In [5]:

```
1 data.info()
```

executed in 16ms, finished 10:19:45 2022-11-23

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8693 entries, 0 to 8692
Data columns (total 13 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   HomePlanet    8693 non-null   object
 1   CryoSleep     8693 non-null   bool
 2   Combi         8590 non-null   object
 3   Cabin3        8590 non-null   object
 4   Destination   8693 non-null   object
 5   Age           8693 non-null   int64
 6   VIP           8693 non-null   bool
 7   RoomService   8693 non-null   int64
 8   FoodCourt     8693 non-null   int64
 9   ShoppingMall  8693 non-null   int64
 10  Spa           8693 non-null   int64
 11  VRDeck        8693 non-null   int64
 12  Transported   8693 non-null   bool
dtypes: bool(3), int64(6), object(4)
memory usage: 704.7+ KB
```

3.2 처리하기 힘든 결측값 제거

In [6]:

```
1 data.isna().sum()
```

executed in 16ms, finished 10:19:45 2022-11-23

Out[6]:

```
HomePlanet      0
CryoSleep       0
Combi         103
Cabin3        103
Destination     0
Age             0
VIP             0
RoomService     0
FoodCourt       0
ShoppingMall    0
Spa             0
VRDeck          0
Transported     0
dtype: int64
```

In [7]:

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

executed in 15ms, finished 10:19:45 2022-11-23

3.3 Boolean 캐스팅

In [8]:

```
# Cabin3의 값을 변환
data['Cabin3'].replace({'P': True,'S': False}, inplace=True)
data['Cabin3'] = data['Cabin3'].astype(bool)
```

executed in 15ms, finished 10:19:45 2022-11-23

3.4 원핫인코딩

In [9]:

```
# 원핫인코딩
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['Combi'])
data=data.drop('Combi',axis=1)
data = data.join(train_encoding)
```

executed in 93ms, finished 10:19:46 2022-11-23

In [10]:

```
data.info()
```

executed in 109ms, finished 10:19:46 2022-11-23

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8590 entries, 0 to 8692
Columns: 4469 entries, CryoSleep to T3
dtypes: bool(4), int64(6), uint8(4459)
memory usage: 37.3 MB
```

3.5 스케일링

In [11]:

```
# 스케일링(Cabin2)
col = ['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 16ms, finished 10:19:46 2022-11-23

In [12]:

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

executed in 31ms, finished 10:19:46 2022-11-23

Out[12]:

	CryoSleep	Cabin3	Age	VIP	RoomService	FoodCourt	ShoppingMall
0	False	True	0.712274	False	-0.333743	-0.280785	-0.282832
1	False	False	-0.332624	False	-0.168530	-0.275148	-0.241196
2	False	False	2.035811	True	-0.268567	1.959032	-0.282832
3	False	False	0.294315	False	-0.333743	0.522818	0.335048
4	False	False	-0.889902	False	0.125518	-0.236941	-0.031350
...	...	...	...	...	...	...	...
8688	False	True	0.851594	True	-0.333743	3.990274	-0.282832
8689	True	False	-0.750583	False	-0.333743	-0.280785	-0.282832
8690	False	False	-0.193304	False	-0.333743	-0.280785	2.834877
8691	False	False	0.224655	False	-0.333743	0.376253	-0.282832
8692	False	False	1.060573	False	-0.142763	2.655529	-0.282832

8590 rows × 4469 columns

In [28]:

```
1 data.columns
```

executed in 15ms, finished 10:45:40 2022-11-23

Out[28]:

```
Index(['CryoSleep', 'Cabin3', 'Age', 'VIP', 'RoomService', 'FoodCour
t',
       'ShoppingMall', 'Spa', 'VRDeck', 'Transported',
       ...
       'G993', 'G994', 'G995', 'G996', 'G998', 'G999', 'T0', 'T1', 'T
2', 'T3'],
      dtype='object', length=4469)
```

4 데이터셋 분리

In [13]:

```
1 X_train, X_test, y_train, y_test = train_test_split(data.drop(['Transported'],
2                                                               data.Transported, random_s
3 X_train, X_val, y_train, y_val = train_test_split(X_train, y_train)
```

executed in 202ms, finished 10:19:46 2022-11-23

5 XGBoost

In [14]:

```
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 15ms, finished 10:19:46 2022-11-23

In [15]:

```
# fmin()에서 호출 시 search_space 값으로 XGBClassifier 교차 검증 학습 후 -1 * roc_a
def bin_objective_func(search_space):
    xgb_clf = xgb.XGBClassifier(n_estimators=100, max_depth=int(search_space['
                                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 16ms, finished 10:19:46 2022-11-23

In [16]:

```
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 11m 27s, finished 10:31:13 2022-11-23

```
[0]     validation_0-auc:0.87531        validation_1-auc:0.83858
[1]     validation_0-auc:0.89466        validation_1-auc:0.85267
[2]     validation_0-auc:0.90672        validation_1-auc:0.86289
[3]     validation_0-auc:0.91147        validation_1-auc:0.86775
[4]     validation_0-auc:0.91668        validation_1-auc:0.86856
[5]     validation_0-auc:0.91862        validation_1-auc:0.86903
[6]     validation_0-auc:0.92302        validation_1-auc:0.87001
[7]     validation_0-auc:0.92567        validation_1-auc:0.87243
[8]     validation_0-auc:0.92760        validation_1-auc:0.87449
[9]     validation_0-auc:0.93018        validation_1-auc:0.87455
[10]    validation_0-auc:0.93093        validation_1-auc:0.87408
[11]    validation_0-auc:0.93110        validation_1-auc:0.87376
[12]    validation_0-auc:0.93180        validation_1-auc:0.87367
[13]    validation_0-auc:0.93199        validation_1-auc:0.87348
[14]    validation_0-auc:0.93266        validation_1-auc:0.87325
[15]    validation_0-auc:0.93386        validation_1-auc:0.87294
[16]    validation_0-auc:0.93582        validation_1-auc:0.87352
[17]    validation_0-auc:0.93682        validation_1-auc:0.87430
[18]    validation_0-auc:0.93822        validation_1-auc:0.87457
```

In [17]:

```
# 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 10:31:13 2022-11-23

In [19]:

```
xgbo = xgb.XGBClassifier(colsample_bytree=0.6167132910325195, gamma=3.25553863
                         learning_rate=0.23322945783668153,
                         max_depth=9, min_child_weight=5, random_state=109)
xgbo.fit(X_train, y_train)
```

executed in 4.71s, finished 10:34:44 2022-11-23

Out[19]:

```
XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1,
              colsample_bytree=0.6167132910325195, early_stopping_roun
ds=None,
              enable_categorical=False, eval_metric=None, feature_type
s=None,
              gamma=3.255538638467651, gpu_id=-1, grow_policy='depthwi
se',
              importance_type=None, interaction_constraints='',
              learning_rate=0.23322945783668153, max_bin=256,
              max_cat_threshold=64, max_cat_to_onehot=4, max_delta_ste
p=0,
              max_depth=9, max_leaves=0, min_child_weight=5, missing=n
an,
              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.

In [20]:

```
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 1.87s, finished 10:34:50 2022-11-23

In [21]:

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

executed in 13ms, finished 10:34:52 2022-11-23

```
오차 행렬
[[1956  438]
 [ 324 2113]]
정확도: 0.8423, 정밀도: 0.8283,     재현율: 0.8670, F1: 0.8472
```

In [22]:

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

executed in 11ms, finished 10:34:54 2022-11-23

```
오차 행렬
[[825 242]
 [219 862]]
정확도: 0.7854, 정밀도: 0.7808,     재현율: 0.7974, F1: 0.7890
```

In [23]:

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

executed in 18ms, finished 10:34:58 2022-11-23

```
오차 행렬
[[620 176]
 [140 675]]
정확도: 0.8038, 정밀도: 0.7932,     재현율: 0.8282, F1: 0.8103
```

In [26]:

```
1 fi = pd.DataFrame(xgbo.feature_importances_)
```

executed in 9ms, finished 10:41:27 2022-11-23

In [27]:

```
1 fi.to_csv('fi.csv')
```

executed in 118ms, finished 10:41:45 2022-11-23

In [ ]:

```
1
```