Contents

1 라이브러리 로딩

In [1]:

```
import numpy as np # Numpy
import pandas as pd # Pandas
import matplotlib as mpl #Matplotlib 세팅
import matplotlib.pyplot as plt # 시각화 .
import seaborn as sns # 시각화 도구
from sklearn.model_selection import trai
from sklearn.model_selection import KFol
from sklearn.cluster import KMeans # 클러
from sklearn.metrics import silhouette_s
import xgboost as xgb # XGBoost
from sklearn.model_selection import GridS
from sklearn.metrics import accuracy_sco
from sklearn.metrics import recall_score
from imblearn.combine import SMOTEENN, SM
from hyperopt import hp, fmin, tpe, Tria

import warnings # 경고문 제거용


%matplotlib inline
%config Inlinebackend.figure_format = 're

# 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.unico
plt.rc('figure', figsize=(10,8))
```

executed in 727ms, finished 11:24:51 2022-11-23

2 데이터 불러오기

In [2]:

```
data = pd.read_excel('train_test_na_fill
```

executed in 1.37s, finished 11:24:53 2022-11-23

3 전처리

Contents

In [3]:

```
data.info()
```

executed in 16ms, finished 11:24:53 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(
memory usage: 1.0+ MB
```

3.1 필요없는 features 제거

In [4]:

```
# 필요없는 features 제거
data.drop(['PassengerId', 'Cabin', 'Cabi
```

executed in 15ms, finished 11:24:53 2022-11-23

In [5]:

```
data.info()
```

executed in 16ms, finished 11:24:53 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   Cabin1        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]:

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

executed in 16ms, finished 11:24:53 2022-11-23

Out[6]:

```
HomePlanet        0
CryoSleep         0
Cabin1          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]:

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

executed in 15ms, finished 11:24:53 2022-11-23

3.3 Boolean 캐스팅

In [8]:

```
# Cabin3의 값을 변환
data['Cabin3'].replace({'P': True,'S': Fa
data['Cabin3'] = data['Cabin3'].astype(bo
```

executed in 16ms, finished 11:24:53 2022-11-23

3.4 원핫인코딩

In [9]:

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

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

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

executed in 15ms, finished 11:24:53 2022-11-23

In [10]:

```
data.info()
```

executed in 14ms, finished 11:24:53 2022-11-23

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8590 entries, 0 to 8692
Data columns (total 24 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   CryoSleep      8590 non-null   bool
 1   Cabin3         8590 non-null   bool
 2   Age            8590 non-null   int64
 3   VIP            8590 non-null   bool
 4   RoomService    8590 non-null   int64
 5   FoodCourt      8590 non-null   int64
 6   ShoppingMall   8590 non-null   int64
 7   Spa            8590 non-null   int64
 8   VRDeck         8590 non-null   int64
 9   Transported    8590 non-null   bool
 10  Earth          8590 non-null   uint8
 11  Europa         8590 non-null   uint8
 12  Mars           8590 non-null   uint8
 13  55 Cancri e    8590 non-null   uint8
```

3.5 스케일링

In [11]:

```
# 스케일링
col = ['Age', 'RoomService','FoodCourt',
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_s
        df[i]=scaled
    return df
```

executed in 15ms, finished 11:24:53 2022-11-23

In [12]:

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

executed in 16ms, finished 11:24:53 2022-11-23

Out[12]:

	CryoSleep	Cabin3	Age	VIP	RoomService
0	False	True	0.712274	False	-0.33374
1	False	False	-0.332624	False	-0.16853
2	False	False	2.035811	True	-0.26856
3	False	False	0.294315	False	-0.33374
4	False	False	-0.889902	False	0.12551
...	...	...	...	...	
8688	False	True	0.851594	True	-0.33374
8689	True	False	-0.750583	False	-0.33374
8690	False	False	-0.193304	False	-0.33374
8691	False	False	0.224655	False	-0.33374
8692	False	False	1.060573	False	-0.14276

8590 rows × 24 columns

In [13]:

```
data.columns
```

executed in 16ms, finished 11:24:53 2022-11-23

Out[13]:

```
Index(['CryoSleep', 'Cabin3', 'Age', 'VIP', 'R
t',
       'ShoppingMall', 'Spa', 'VRDeck', 'Trans
a',
       'Mars', '55 Cancri e', 'PSO J318.5-22',
'B', 'C',
       'D', 'E', 'F', 'G', 'T'],
      dtype='object')
```

4 데이터셋 분리

In [14]:

```
X_train, X_test, y_train, y_test = train_

X_train, X_val, y_train, y_val = train_te
```

executed in 16ms, finished 11:24:53 2022-11-23

5 XGBoost

In [15]:

```
xgb_search_space = {'max_depth': hp.quni
                    'min_child_weight': hp
                    'colsample_bytree': hp
                    'learning_rate': hp.u
                    'gamma': hp.uniform('
```

executed in 16ms, finished 11:24:53 2022-11-23

In [16]:

```
# fmin()에서 호출 시 search_space 값으로 XG
def bin_objective_func(search_space):
    xgb_clf = xgb.XGBClassifier(n_estimat
                                min_child_we
                                colsample_byt
                                learning_rate
                                gamma=search_

    # 3개 k-fold 방식으로 평가된 roc_auc 지
    roc_auc_list = []

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

    # X_train을 다시 학습과 검증용 데이터로 
    for tr_index, val_index in kf.split(X
        # kf.split(X_train)으로 추출된 학습
        X_tr, y_tr = X_train.iloc[tr_inde
        X_val, y_val = X_train.iloc[val_

        # early stopping은 30회로 설정하고
        xgb_clf.fit(X_tr, y_tr, early_sto
                    eval_set=[(X_tr, y_tr

        # 1로 예측한 확률값 추출 후 roc auc
        score = roc_auc_score(y_val, xgb_
        roc_auc_list.append(score)

    # 3개 k-fold로 계산된 roc_auc 값의 평균
    # HyperOPT는 목적함수의 최솟값을 위한 입
    return -1*np.mean(roc_auc_list)
```

executed in 15ms, finished 11:24:53 2022-11-23

In [17]:

```
trials = Trials()

# fmin() 함수를 호출. max_evals 지정된 횟수
best = fmin(fn=bin_objective_func,
            space=xgb_search_space,
            algo=tpe.suggest,
            max_evals=50, # 최대 반복 횟수를
            trials=trials, rstate=np.rand

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

executed in 32.6s, finished 11:25:25 2022-11-23

```
[0]     validation_0-auc:0.88826        valida
[1]     validation_0-auc:0.90424        valida
[2]     validation_0-auc:0.91625        valida
[3]     validation_0-auc:0.92169        valida
[4]     validation_0-auc:0.92532        valida
[5]     validation_0-auc:0.92815        valida
[6]     validation_0-auc:0.93043        valida
[7]     validation_0-auc:0.93155        valida
[8]     validation_0-auc:0.93388        valida
[9]     validation_0-auc:0.93488        valida
[10]    validation_0-auc:0.93587        valida
[11]    validation_0-auc:0.94027        valida
[12]    validation_0-auc:0.94098        valida
[13]    validation_0-auc:0.94329        valida
[14]    validation_0-auc:0.94341        valida
[15]    validation_0-auc:0.94468        valida
[16]    validation_0-auc:0.94523        valida
[17]    validation_0-auc:0.94558        valida
[18]    validation_0-auc:0.94712        valida
```

In [18]:

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

    print('오차 행렬')
    print(confusion)

    print('정확도: {0:.4f}, 정밀도: {1:.4f}
    재현율: {2:.4f}, F1: {3:.4f}'.format(
```

executed in 16ms, finished 11:25:25 2022-11-23

In [26]:

```
xgbo = xgb.XGBClassifier(colsample_bytree
                         learning_rate=0
                         max_depth=9, mi
xgbo.fit(X_train, y_train)
```

executed in 124ms, finished 11:26:17 2022-11-23

Out[26]:

```
XGBClassifier(base_score=0.5, booster='gbtree'
              colsample_bylevel=1, colsample_b
              colsample_bytree=0.9227415127512
ds=None,
              enable_categorical=False, eval_m
s=None,
              gamma=3.9860828876917274, gpu_id
ise',
              importance_type=None, interactio
              learning_rate=0.0679559959769339
              max_cat_threshold=64, max_cat_to
p=0,
              max_depth=9, max_leaves=0, min_c
an,
              monotone_constraints='()', n_est
              num_parallel_tree=1, predictor='
...)
```

In a Jupyter environment, please rerun this cell to show th the notebook.
On GitHub, the HTML representation is unable to render, p with nbviewer.org.

In [27]:

```
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 24ms, finished 11:26:21 2022-11-23

In [28]:

```
get_clf_eval(y_train, train_pred, train_
```

executed in 20ms, finished 11:26:22 2022-11-23

```
오차 행렬
[[1976  430]
 [ 344 2081]]
정확도: 0.8398, 정밀도: 0.8288,     재현율: 0.858
```

In [29]:

```
get_clf_eval(y_test, test_pred, test_prol
```

executed in 17ms, finished 11:26:24 2022-11-23

```
오차 행렬
[[838 229]
 [212 869]]
정확도: 0.7947, 정밀도: 0.7914,     재현율: 0.803
```

In [30]:

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

executed in 13ms, finished 11:26:25 2022-11-23

```
오차 행렬
[[601 183]
 [139 688]]
정확도: 0.8001, 정밀도: 0.7899,     재현율: 0.831
```

In [33]:

```
fi = pd.DataFrame(xgbo.feature_importance
```

executed in 13ms, finished 11:28:08 2022-11-23

In [34]:

```
fi.to_csv('fi_3.csv')
```

executed in 8ms, finished 11:28:09 2022-11-23

In [ ]:

```

```