머신러닝 파이프라인

학습 목표

- 머신러닝 파이프라인(pipeline)과 학습한 모델을 저장과 활용하는 방법을 알아본다.

목차

01 데이터 불러오기

목차로 이동하기

In [35]:

```
import pandas as pd
from sklearn.model_selection import train_test_split

train = pd.read_csv("../dataset/Space_Titanic/train.csv")
test = pd.read_csv("../dataset/Space_Titanic/test.csv")
sub = pd.read_csv("../dataset/Space_Titanic/sample_submission.csv")

train.shape, test.shape, sub.shape
```

Out[35]:

```
((8693, 14), (4277, 13), (4277, 2))
```

In [36]:

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8693 entries, 0 to 8692
Data columns (total 14 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   PassengerId   8693 non-null   object 
 1   HomePlanet    8492 non-null   object 
 2   CryoSleep     8476 non-null   object 
 3   Cabin         8494 non-null   object 
 4   Destination   8511 non-null   object 
 5   Age           8514 non-null   float64
 6   VIP           8490 non-null   object 
 7   RoomService   8512 non-null   float64
 8   FoodCourt     8510 non-null   float64
 9   ShoppingMall  8485 non-null   float64
 10  Spa           8510 non-null   float64
 11  VRDeck        8505 non-null   float64
 12  Name          8493 non-null   object 
 13  Transported   8693 non-null   bool   
dtypes: bool(1), float64(6), object(7)
memory usage: 891.5+ KB
```

In [37]:

```
train.head()
```

Out[37]:

	PassengerId	HomePlanet	CryoSleep	Cabin	Destination	Age	VIP	RoomService	FoodCo
0	0001_01	Europa	False	B/0/P	TRAPPIST-1e	39.0	False	0.0	
1	0002_01	Earth	False	F/0/S	TRAPPIST-1e	24.0	False	109.0	
2	0003_01	Europa	False	A/0/S	TRAPPIST-1e	58.0	True	43.0	357
3	0003_02	Europa	False	A/0/S	TRAPPIST-1e	33.0	False	0.0	128
4	0004_01	Earth	False	F/1/S	TRAPPIST-1e	16.0	False	303.0	7

In [38]:

```
train.columns
```

Out[38]:

```
Index(['PassengerId', 'HomePlanet', 'CryoSleep', 'Cabin', 'Destination', 'Age',
       'VIP', 'RoomService', 'FoodCourt', 'ShoppingMall', 'Spa', 'VRDeck',
       'Name', 'Transported'],
      dtype='object')
```

In [39]:

```
sel = [ 'RoomService', 'FoodCourt', 'ShoppingMall', 'Spa', 'VRDeck' ]
X = train[sel]
y = train['Transported']

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
```

파이프라인에 사용될 임퓨터, 스케일러, 분류 모델 정의

In [40]:

```
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import MinMaxScaler
from sklearn.tree import DecisionTreeClassifier as DTC
```

In [41]:

```
imputer = SimpleImputer(strategy = "mean")
scaler = MinMaxScaler()
model = DTC()
```

02 파이프라인 생성 및 학습

목차로 이동하기

- 각 요소의 이름은 imputer, scaler, model 로 지정.

In [42]:

```
from sklearn.pipeline import Pipeline

pipe_line = Pipeline([ ("imputer", imputer), ("scaler", scaler), ("model", model) ])
pipe_line.fit(X_train, y_train)
```

Out[42]:

```
Pipeline(steps=[('imputer', SimpleImputer()), ('scaler', MinMaxScaler()),
                ('model', DecisionTreeClassifier())])
```

파이프라인을 이용한 예측 수행

In [43]:

```
pipe_line.predict(X_test)
```

Out[43]:

```
array([False,  True, False, ...,  True,  True,  True])
```

정리

- 일부 변수를 선택 후, 결측치는 평균으로 대체된 후, 스케일링된 X_test가 DecisionTree 모델에 입력되어 나온 결과이다.
- predict 메서드는 파이프라인의 마지막 요소인 SVC 모델의 메서드를 사용하였다.

Pipeline 클래스의 단점

- 손쉽게 파이프라인을 구축할 수 있는 장점이 있다.
- 단, 다음과 같은 문제도 존재한다.
  - 1. 사이킷런 인스턴스가 아닌 다른 클래스의 인스턴스가 파이프라인에 포함되면 정상적으로 작동하지 않을 수 있음.
  - 2. 하이퍼파라미터 튜닝 등을 할 때 전체 파이프라인을 계속해서 수정해야 한다.
    - 해결 제안 : 함수나 클래스를 사용하여 파이프라인을 함수화 해보기

03 파이프라인 함수로 만들기

목차로 이동하기

pipe_line_fnc 함수 만들기

In [44]:

```
def pipe_line_fnc(X, imputer, scaler, model):
    X = imputer.transform(X)
    X = scaler.transform(X)
    pred = model.predict(X)
    return pred

pred_Y = pipe_line_fnc(X_test, imputer, scaler, model)

pred_Y[:5]
```

Out[44]:

```
array([False,  True, False, False,  True])
```

In [45]:

```
pred_Y.shape
```

Out[45]:

```
(2174,)
```

In [46]:

```
from sklearn.metrics import accuracy_score
accuracy_score(pred_Y, y_test)
```

Out[46]:

```
0.7396504139834407
```

최종 결과 73.59%

In [47]:

```
from sklearn.metrics import classification_report
print( classification_report(pred_Y, y_test) )
```

```
              precision    recall  f1-score   support

       False       0.63      0.80      0.71       849
        True       0.85      0.70      0.77      1325

    accuracy                           0.74      2174
   macro avg       0.74      0.75      0.74      2174
weighted avg       0.76      0.74      0.74      2174
```

- 정밀도 : 0.85, 재현율 : 0.70, f1-score : 0.76

04 학습내용 저장 및 불러오기

목차로 이동하기

In [50]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.pipeline import Pipeline
# from sklearn.externals import joblib
import sklearn.externals
import joblib
```

In [51]:

```
sel = [ 'RoomService', 'FoodCourt', 'ShoppingMall', 'Spa', 'VRDeck' ]
X = train[sel]
y = train['Transported']

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
```

In [52]:

```
imputer = SimpleImputer(strategy = "mean")
scaler = MinMaxScaler()
model = LogisticRegression()
```

In [53]:

```
pipe_line_Log = Pipeline([ ("imputer", imputer), ("scaler", scaler), ("model", model) ])
pipe_line_Log.fit(X_train, y_train)
pred = pipe_line_Log.predict(X_test)
pred[0:5]
```

Out[53]:

```
array([ True,  True, False, False,  True])
```

In [54]:

```
# 정확도 확인
print( accuracy_score(pred, y_test) )
```

0.7382704691812327

In [55]:

```
model = KNeighborsClassifier()
```

In [56]:

```
pipe_line_knn = Pipeline([ ("imputer", imputer), ("scaler", scaler), ("model", model) ])
pipe_line_knn.fit(X_train, y_train)
pred = pipe_line_knn.predict(X_test)

# 정확도 확인
print( accuracy_score(pred, y_test) )
```

0.7723091076356946

In [59]:

```
joblib.dump(pipe_line_knn, "../dataset/Space_Titanic/model_pipe_knn.joblib" )
```

Out[59]:

```
['../dataset/Space_Titanic/model_pipe_knn.joblib']
```

In [60]:

```
import os
os.listdir("../dataset/Space_Titanic/")
```

Out[60]:

```
['first_sub.csv',
 'model_pipe_knn.joblib',
 'sample_submission.csv',
 'test.csv',
 'train.csv']
```

불러오기 후, 확인

In [61]:

```
import sklearn.externals
import joblib

pipe_knn = joblib.load("../dataset/Space_Titanic/model_pipe_knn.joblib")

pred = pipe_knn.predict(X_test)

# 정확도 확인
print( accuracy_score(pred, y_test) )
```

0.7723091076356946