[Spaceship Titanic 데이터 분석]

- 우주선에서 승객을 보낼때, 다른 차원 이송되었는지, 아닌지 예측하기
- 데이터 출처 : https://www.kaggle.com/competitions/spaceship-titanic/data (https://www.kaggle.com/competitions/spaceship-titanic/data)
- 데이터 분석 코드
  - github 코드 (https://github.com/LDJWJ/dataAnalysis/blob/main/01_04A_Spaceship_Titanic.ipynb)
  - HTML코드 (https://ldjwj.github.io/dataAnalysis/01_04A_Spaceship_Titanic.html)

대회 개요

- 우주선 타이타닉의 승객이 다른 차원으로 이송되었는지(True-1) 아닌지(False-0)

학습 내용

- 데이터에 대한 기본 탐색을 수행해 봅니다.
- 데이터 결측치에 대한 처리를 수행해 봅니다.
- 베이스 라인 모델을 만들어봅니다.

목차

01. 데이터 설명

목차로 이동하기

Data 설명

구분	설명	비고
PassengerId	각 승객에 대한 고유 ID 각 ID는 승객이 함께 여행하는 그룹 gggg_pp을 gggg나타내며 그룹 pp내 번호	그룹의 사람들은 종종 가족 구성원이지만 항상 그런 것은 아닙니다.
HomePlanet	승객이 출발한 행성, 일반적으로 영구 거주 행성입니다.	'Europa', 'Earth', 'Mars'
CryoSleep	승객이 항해 기간 동안 정지된 애니메이션에 놓이도록 선택했는지 여부.	False, True
Cabin	승객이 머물고 있는 객실 번호	
Destination	승객이 출발할 행성	'TRAPPIST-1e', 'PSO J318.5-22', '55 Cancri e'

구분	설명	비고
Age	승객의 나이	
VIP	승객이 항해 중 특별 VIP 서비스를 지불했는지 여부	
RoomService, FoodCourt, ShoppingMall	승객이 우주선 타이타닉 의 다양한 고급 편의 시설에 대해 청구한금액	
Name	승객의 성과 이름	
Transported	승객이 다른 차원으로 이송되었는지 여부.	예측 대상 열

데이터 셋

- train.csv - 학습용 데이터 셋.
- test.csv - Transported을 예측하는 것.
- sample_submission.csv - 올바른 형식의 제출 파일
  - PassengerId - 테스트 세트의 각 승객에 대한 ID
  - Transported - 목표. 각 승객이 다른 차원 이송되었는지 여부

02. 라이브러리 및 데이터 불러오기

목차로 이동하기

라이브러리 불러오기

In [91]:

```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
```

데이터 불러오기

In [92]:

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

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

Out[92]:

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

- 학습용 데이터 셋 : 8693행 14열

- 테스트용 데이터 셋 : 4277행 13열

In [93]:

```
print( train.columns, end="₩n₩n")
print( test.columns)
```

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

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

03. 데이터 탐색

목차로 이동하기

예측할 피처(특징)은 Transported

Transported은 어떠한 값을 갖을까?

In [94]:

```
train['Transported'].unique()
```

Out[94]:

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

In [95]:

```
test['Transported'].unique()  # test에는 예측할 컬럼이 없음. 에러 발생
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File ~\anaconda3\lib\site-packages\pandas\core\indexes\base.py:3621, in Index.get_loc(self, key, method, tolerance)
   3620 try:
-> 3621     return self._engine.get_loc(casted_key)
   3622 except KeyError as err:

File ~\anaconda3\lib\site-packages\pandas\_libs\index.pyx:136, in pandas._libs.index.IndexEngine.get_loc()

File ~\anaconda3\lib\site-packages\pandas\_libs\index.pyx:163, in pandas._libs.index.IndexEngine.get_loc()

File pandas\_libs\hashtable_class_helper.pxi:5198, in pandas._libs.hashtable.PyObjectHashTable.get_item()

File pandas\_libs\hashtable_class_helper.pxi:5206, in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'Transported'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Input In [95], in <cell line: 1>()
----> 1 test['Transported'].unique()

File ~\anaconda3\lib\site-packages\pandas\core\frame.py:3505, in DataFrame.__getitem__(self, key)
   3503 if self.columns.nlevels > 1:
   3504     return self._getitem_multilevel(key)
-> 3505 indexer = self.columns.get_loc(key)
   3506 if is_integer(indexer):
   3507     indexer = [indexer]

File ~\anaconda3\lib\site-packages\pandas\core\indexes\base.py:3623, in Index.get_loc(self, key, method, tolerance)
   3621     return self._engine.get_loc(casted_key)
   3622 except KeyError as err:
-> 3623     raise KeyError(key) from err
   3624 except TypeError:
   3625     # If we have a listlike key, _check_indexing_error will raise
   3626     #  InvalidIndexError. Otherwise we fall through and re-raise
   3627     #  the TypeError.
   3628     self._check_indexing_error(key)

KeyError: 'Transported'
```

In [96]:

```
train.columns
```

Out[96]:

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

In [97]:

```
for one in train.columns:
    print("컬럼명 : ", one)
    print("유일한 값의 길이 및 값 : ", len( train[one].unique() ), train[one].unique(), end="₩n₩n")
```

```
컬럼명 :  PassengerId
유일한 값의 길이 및 값 :  8693 ['0001_01' '0002_01' '0003_01' ... '9279_01' '9280_0
1' '9280_02']

컬럼명 :  HomePlanet
유일한 값의 길이 및 값 :  4 ['Europa' 'Earth' 'Mars' nan]

컬럼명 :  CryoSleep
유일한 값의 길이 및 값 :  3 [False True nan]

컬럼명 :  Cabin
유일한 값의 길이 및 값 :  6561 ['B/0/P' 'F/0/S' 'A/0/S' ... 'G/1499/S' 'G/1500/S'
'E/608/S']

컬럼명 :  Destination
유일한 값의 길이 및 값 :  4 ['TRAPPIST-1e' 'PSO J318.5-22' '55 Cancri e' nan]

컬럼명 :  Age
유일한 값의 길이 및 값 :  81 [39. 24. 58. 33. 16. 44. 26. 28. 35. 14. 34. 45. 32. 4
8. 31. 27.  0.  1.
 49. 29. 10.  7. 21. 62. 15. 43. 47.  2. 20. 23. 30. 17. 55.  4. 19. 56.
 nan 25. 38. 36. 22. 18. 42. 37. 13.  8. 40.  3. 54.  9.  6. 64. 67. 61.
 50. 41. 57. 11. 52. 51. 46. 60. 63. 59.  5. 79. 68. 74. 12. 53. 65. 71.
 75. 70. 76. 78. 73. 66. 69. 72. 77.]

컬럼명 :  VIP
유일한 값의 길이 및 값 :  3 [False True nan]

컬럼명 :  RoomService
유일한 값의 길이 및 값 :  1274 [   0.  109.   43. ... 1569. 8586.  745.]

컬럼명 :  FoodCourt
유일한 값의 길이 및 값 :  1508 [   0.    9. 3576. ... 3208. 6819. 4688.]

컬럼명 :  ShoppingMall
유일한 값의 길이 및 값 :  1116 [   0.   25.  371. ... 1085.  510. 1872.]

컬럼명 :  Spa
유일한 값의 길이 및 값 :  1328 [   0.  549. 6715. ... 2868. 1107. 1643.]

컬럼명 :  VRDeck
유일한 값의 길이 및 값 :  1307 [   0.   44.   49. ... 1164.  971. 3235.]

컬럼명 :  Name
유일한 값의 길이 및 값 :  8474 ['Maham Ofracculy' 'Juanna Vines' 'Altark Susent' ...
'Fayey Connon'
 'Celeon Hontichre' 'Propsh Hontichre']

컬럼명 :  Transported
유일한 값의 길이 및 값 :  2 [False  True]
```

01 주어진 데이터(학습) 데이터를 나눈다. (학습용, 자체 평가용)

02 모델을 선택 및 학습하고, 이를 토대로 자체 평가를 토대로 모델 최종 선택

03 마지막 선택된 모델로 test의 Transported를 예측하고 제출

04. 데이터 처리

목차로 이동하기

In [98]:

```
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
```

- 결측치 존재하는 컬럼 : HomePlanet, CryoSleep, Cabin, Destination, Age, VIP, RoomService, FoodCourt, ShoppingMall, Spa, VRDeck, Name
- 결측치 존재하지 않는 컬럼 : PassengerId, Transported

In [99]:

```
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4277 entries, 0 to 4276
Data columns (total 13 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   PassengerId   4277 non-null   object 
 1   HomePlanet    4190 non-null   object 
 2   CryoSleep     4184 non-null   object 
 3   Cabin         4177 non-null   object 
 4   Destination   4185 non-null   object 
 5   Age           4186 non-null   float64
 6   VIP           4184 non-null   object 
 7   RoomService   4195 non-null   float64
 8   FoodCourt     4171 non-null   float64
 9   ShoppingMall  4179 non-null   float64
 10  Spa           4176 non-null   float64
 11  VRDeck        4197 non-null   float64
 12  Name          4183 non-null   object 
dtypes: float64(6), object(7)
memory usage: 434.5+ KB
```

In [100]:

```
test.isnull().sum()
```

Out[100]:

```
PassengerId     0
HomePlanet     87
CryoSleep      93
Cabin         100
Destination    92
Age            91
VIP            93
RoomService    82
FoodCourt     106
ShoppingMall   98
Spa           101
VRDeck         80
Name           94
dtype: int64
```

- 결측치 존재하는 컬럼 : HomePlanet, CryoSleep, Cabin, Destination, Age, VIP, RoomService, FoodCourt, ShoppingMall, Spa, VRDeck, Name
- 결측치 존재하지 않는 컬럼 : PassengerId

결측치는 전부 있다. 따라서 이에 대한 결측치 처리가 필요하다.

In [101]:

```
train.head()
```

Out[101]:

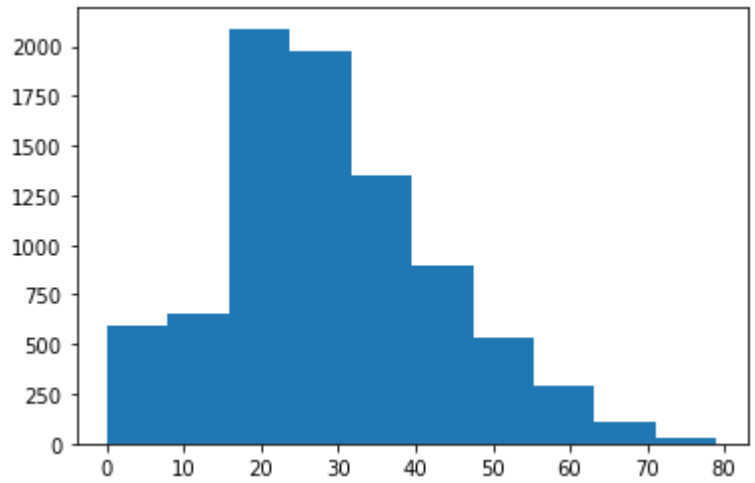
	PassengerId	HomePlanet	CryoSleep	Cabin	Destination	Age	VIP	RoomService	FoodCo
0	0001_01	Europa	False	B/0/P	TRAPPIST-1e	39.0	False	0.0	
1	0002_01	Earth	False	F/0/S	TRAPPIST-1e	24.0	False	109.0	
2	0003_01	Europa	False	A/0/S	TRAPPIST-1e	58.0	True	43.0	357
3	0003_02	Europa	False	A/0/S	TRAPPIST-1e	33.0	False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

In [102]:

```
plt.hist(train['Age'])
```

Out[102]:

```
(array([ 591.,  649., 2089., 1976., 1350.,  893.,  536.,  294.,  106.,
         30.]),
 array([ 0. ,  7.9, 15.8, 23.7, 31.6, 39.5, 47.4, 55.3, 63.2, 71.1, 79. ]),
 <BarContainer object of 10 artists>)
```



(실습) 다른 특징들의 데이터 분포 확인해 보기

숫자이면서 결측치가 다 있음. 최소한의 빠른 모델을 만들기 위해 몇개만 결측치를 처리한다.

sel = ['Age', 'RoomService', 'FoodCourt']

- 어떠한 값으로 결측치 처리를 할 것인가? 다양한 논의가 있다. 여기에서는 중앙값으로 결측치 처리를 수행.

- 'Age', 'RoomService', 'FoodCourt'를 전부 중앙값으로 결측치 처리하자.

In [103]:

```
# 비어 있는 행 선택 후, 컬럼은 'RoomService' 선택 후, 중앙값으로 넣기
train.loc[ train['Age'].isnull() , 'Age' ] = train['Age'].median()
train.loc[ train['RoomService'].isnull() , 'RoomService' ] = train['RoomService'].median()
train.loc[ train['FoodCourt'].isnull() , 'FoodCourt' ] = train['FoodCourt'].median()

# 비어 있는지 확인
train.isnull().sum()
```

Out[103]:

```
PassengerId       0
HomePlanet      201
CryoSleep       217
Cabin           199
Destination     182
Age               0
VIP             203
RoomService       0
FoodCourt         0
ShoppingMall    208
Spa             183
VRDeck          188
Name            200
Transported       0
dtype: int64
```

실습 : 테스트 셋에 대해서도 처리를 수행해 보자.

test 데이터 셋도 동일하게 처리

In [104]:

```
# 비어 있는 행 선택 후, 컬럼은 'RoomService' 선택 후, 중앙값으로 넣기
test.loc[ test['Age'].isnull() , 'Age' ] = test['Age'].median()
test.loc[ test['RoomService'].isnull() , 'RoomService' ] = test['RoomService'].median()
test.loc[ test['FoodCourt'].isnull() , 'FoodCourt' ] = test['FoodCourt'].median()

# 비어 있는지 확인
test.isnull().sum()
```

Out[104]:

```
PassengerId       0
HomePlanet       87
CryoSleep        93
Cabin           100
Destination      92
Age               0
VIP              93
RoomService       0
FoodCourt         0
ShoppingMall     98
Spa             101
VRDeck           80
Name             94
dtype: int64
```

In [105]:

```
train['Transported'].unique()
```

Out[105]:

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

In [106]:

```
train.head()
```

Out[106]:

	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 [107]:

```
print(train.Transported.unique())
train.head()
```

```
[False  True]
```

Out[107]:

	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

우선 결측치 처리된 컬럼을 이용해 보기.

In [108]:

```
sel = ['Age', 'RoomService', 'FoodCourt']

X = train[sel]
y = train['Transported']

last_test = test[sel]

X
```

Out[108]:

	Age	RoomService	FoodCourt
0	39.0	0.0	0.0
1	24.0	109.0	9.0
2	58.0	43.0	3576.0
3	33.0	0.0	1283.0
4	16.0	303.0	70.0
...	...	...	...
8688	41.0	0.0	6819.0
8689	18.0	0.0	0.0
8690	26.0	0.0	0.0
8691	32.0	0.0	1049.0
8692	44.0	126.0	4688.0

8693 rows × 3 columns

In [109]:

```
y
```

Out[109]:

```
0       False
1        True
2       False
3       False
4        True
        ...
8688    False
8689    False
8690     True
8691    False
8692     True
Name: Transported, Length: 8693, dtype: bool
```

데이터 나누기

In [110]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.3,
                                                    random_state = 0)

X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[110]:

```
((6085, 3), (2608, 3), (6085,), (2608,))
```

05. 첫번째 모델 만들어보기

목차로 이동하기

In [111]:

```
X_train.info(), X_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6085 entries, 7289 to 2732
Data columns (total 3 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Age          6085 non-null   float64
 1   RoomService  6085 non-null   float64
 2   FoodCourt    6085 non-null   float64
dtypes: float64(3)
memory usage: 190.2 KB
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2608 entries, 3601 to 5117
Data columns (total 3 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Age          2608 non-null   float64
 1   RoomService  2608 non-null   float64
 2   FoodCourt    2608 non-null   float64
dtypes: float64(3)
memory usage: 81.5 KB
```

Out[111]:

```
(None, None)
```

In [112]:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
```

In [113]:

```
model1 = KNeighborsClassifier()
model1.fit(X_train, y_train)
print("학습용 정확도 : ", model1.score(X_train,y_train) )
print("테스트용 정확도 : ", model1.score(X_test,y_test) )
```

학습용 정확도 : 0.6979457682826623
테스트용 정확도 : 0.6621932515337423

In [114]:

```
model2 = DecisionTreeClassifier(max_depth=3, random_state=0)
model2.fit(X_train, y_train)
print("학습용 정확도 : ", model2.score(X_train,y_train) )
print("테스트용 정확도 : ", model2.score(X_test,y_test) )
```

학습용 정확도 : 0.7250616269515201
테스트용 정확도 : 0.7212423312883436

In [115]:

```
model3 = RandomForestClassifier(max_depth=3, random_state=0)
model3.fit(X_train, y_train)
print("학습용 정확도 : ", model3.score(X_train,y_train) )
print("테스트용 정확도 : ", model3.score(X_test,y_test) )
```

학습용 정확도 : 0.7175020542317173
테스트용 정확도 : 0.7051380368098159

DecisionTreeClassifier 모델이 좋아보임. 우선은 이걸로 최종 모델을 해 본다.
max_depth는 최적으로 맞춰본다.

In [116]:

```
depth_num = range(1, 15, 1)

for num in depth_num:
    model1 = DecisionTreeClassifier(max_depth= num, random_state=0)
    model1.fit(X_train, y_train)

    print("max_depth 값 : ", num)
    print("학습용 정확도 : ", model1.score(X_train,y_train) )
    print("테스트용 정확도 : ", model1.score(X_test,y_test) )
```

```
max_depth 값 :  1
학습용 정확도 :  0.667378800328677
테스트용 정확도 :  0.6618098159509203
max_depth 값 :  2
학습용 정확도 :  0.7117502054231717
테스트용 정확도 :  0.7024539877300614
max_depth 값 :  3
학습용 정확도 :  0.7250616269515201
테스트용 정확도 :  0.7212423312883436
max_depth 값 :  4
학습용 정확도 :  0.7262119967132292
테스트용 정확도 :  0.7158742331288344
max_depth 값 :  5
학습용 정확도 :  0.7296631059983566
테스트용 정확도 :  0.718558282208589
max_depth 값 :  6
학습용 정확도 :  0.7336072308956451
테스트용 정확도 :  0.7105061349693251
max_depth 값 :  7
학습용 정확도 :  0.7377156943303205
테스트용 정확도 :  0.7162576687116564
max_depth 값 :  8
학습용 정확도 :  0.743303204601479
테스트용 정확도 :  0.7059049079754601
max_depth 값 :  9
학습용 정확도 :  0.7515201314708299
테스트용 정확도 :  0.7039877300613497
max_depth 값 :  10
학습용 정확도 :  0.7600657354149548
테스트용 정확도 :  0.7020705521472392
max_depth 값 :  11
학습용 정확도 :  0.7692686935086278
테스트용 정확도 :  0.6951687116564417
max_depth 값 :  12
학습용 정확도 :  0.7789646672144618
테스트용 정확도 :  0.6932515337423313
max_depth 값 :  13
학습용 정확도 :  0.7878389482333608
테스트용 정확도 :  0.6913343558282209
max_depth 값 :  14
학습용 정확도 :  0.7973705834018078
테스트용 정확도 :  0.6859662576687117
```

- max_depth=8일때가 가장 좋음. 최종 모델을 max_depth=8로 한다.

In [117]:

```
last_model = DecisionTreeClassifier(max_depth= num, random_state=0)
last_model.fit(X_train, y_train)
pred = last_model.predict(last_test)
pred[0:10]
```

Out[117]:

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

In [118]:

```
### 제출용 파일 생성
sub.columns
```

Out[118]:

```
Index(['PassengerId', 'Transported'], dtype='object')
```

In [119]:

```
sub['Transported'] = pred
sub.to_csv("./data/Space_Titanic/first_sub_2209.csv", index=False)
```

- 베이스라인 모델 점수 : Score: 0.69277

실습 : 나머지 특징(피처)에 대해서도 수행해 보기

- 'ShoppingMall', 'Spa', 'VRDeck'