01. 데이터 준비 및 라이브러리 임포트

In [1]:
```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt

from sklearn.model_selection import StratifiedKFold, train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.impute import SimpleImputer
from sklearn.metrics import accuracy_score

import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
```

In [2]:
```
import time
import warnings
warnings.filterwarnings('ignore')
```

데이터 불러오기

In [3]:
```
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[3]: ((8693, 14), (4277, 13), (4277, 2))

학습용 데이터 셋 : 8693개
테스트용 데이터 셋 : 4277개

In [4]:
```
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')
```

예측할 피처(특징)은 **Transported**

Transported은 어떠한 값을 갖을까?

In [5]:
```
train['Transported'].unique()
```

Out[5]:
```
array([False,  True])
```

결측치 확인

In [6]:
```
train.isnull().sum()
```

Out[6]:
```
PassengerId       0
HomePlanet      201
CryoSleep       217
Cabin           199
Destination     182
Age             179
VIP             203
RoomService     181
FoodCourt       183
ShoppingMall    208
Spa             183
VRDeck          188
Name            200
Transported       0
dtype: int64
```

In [7]:
```
test.isnull().sum()
```

Out[7]:
```
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
```

각 변수의 기본 통계량 확인

In [8]:
```
train.describe()
```

Out[8]:

	Age	RoomService	FoodCourt	ShoppingMall	Spa	VRDeck
count	8514.000000	8512.000000	8510.000000	8485.000000	8510.000000	8505.000000
mean	28.827930	224.687617	458.077203	173.729169	311.138778	304.854791
std	14.489021	666.717663	1611.489240	604.696458	1136.705535	1145.717189
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	19.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	27.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	38.000000	47.000000	76.000000	27.000000	59.000000	46.000000
max	79.000000	14327.000000	29813.000000	23492.000000	22408.000000	24133.000000

In [9]:
```
test.describe()
```

Out[9]:

	Age	RoomService	FoodCourt	ShoppingMall	Spa	VRDeck
count	4186.000000	4195.000000	4171.000000	4179.000000	4176.000000	4197.000000
mean	28.658146	219.266269	439.484296	177.295525	303.052443	310.710031
std	14.179072	607.011289	1527.663045	560.821123	1117.186015	1246.994742
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	19.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	26.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	37.000000	53.000000	78.000000	33.000000	50.000000	36.000000
max	79.000000	11567.000000	25273.000000	8292.000000	19844.000000	22272.000000

02. 데이터 살펴보기(EDA)

In [10]:
```
### PassengerId 삭제 및 특징(feature)명

train_n = train.drop(["PassengerId"] , axis = 1)
test_n = test.drop(["PassengerId"] , axis = 1)

TARGET = 'Transported'
FEATURES = [col for col in train_n.columns if col != TARGET]
FEATURES
```

Out[10]:
```
['HomePlanet',
 'CryoSleep',
 'Cabin',
 'Destination',
 'Age',
 'VIP',
 'RoomService',
 'FoodCourt',
 'ShoppingMall',
 'Spa',
 'VRDeck',
 'Name']
```

Feature 살펴보기

In [11]:
```
# 각 feature의 중복을 제외한 유일한 값의 갯수
print(FEATURES)
df = pd.concat([train[FEATURES], test[FEATURES]], axis=0)

# 일부 특징 값 확인.
print( df["HomePlanet"].nunique() )
print( df["CryoSleep"].nunique() )
print( df["Cabin"].nunique() )
```

```
['HomePlanet', 'CryoSleep', 'Cabin', 'Destination', 'Age', 'VIP', 'RoomService', 'Food
Court', 'ShoppingMall', 'Spa', 'VRDeck', 'Name']
3
2
9825
```

텍스트(text), 범주형(category), 연속형(continuous) 자료로 분류

In [12]:
```
# 변수별 feature 분류
df = pd.concat([train[FEATURES], test[FEATURES]], axis=0)

text_features = ["Cabin", "Name"]
cat_features = [col for col in FEATURES
                        if df[col].nunique() < 25 and col not in
text_features]
cont_features = [col for col in FEATURES
                        if df[col].nunique() >= 25 and col not in
text_features]
```

In [13]:
```
print( text_features )
print( cat_features )
print( cont_features )
```

```
['Cabin', 'Name']
['HomePlanet', 'CryoSleep', 'Destination', 'VIP']
['Age', 'RoomService', 'FoodCourt', 'ShoppingMall', 'Spa', 'VRDeck']
```

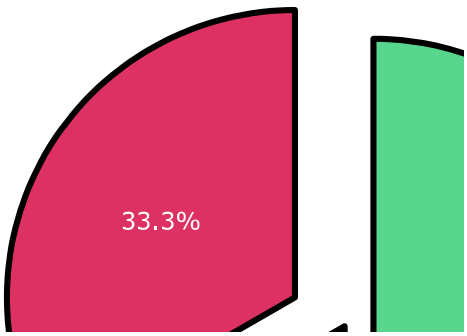
In [14]:
```
del df

print(f'Total number of features: {len(FEATURES)}')
print(f'Number of categorical features: {len(cat_features)}')
print(f'Number of continuos features: {len(cont_features)}')
print(f'Number of text features: {len(text_features)}')
```

```
Total number of features: 12
Number of categorical features: 4
Number of continuos features: 6
Number of text features: 2
```

feature별 분류

In [15]:
```
labels=['Categorical', 'Continuous', "Text"]  # 범주형, 연속형, 텍스트
values= [len(cat_features), len(cont_features), len(text_features)]
colors = ['#DE3163', '#58D68D']

fig = go.Figure(data=[go.Pie(
                labels=labels,
                values=values,
                pull=[0.2, 0.1, 0 ],   # 중심에서 띄우는 정도
```

```
                marker=dict(colors=colors,
                line=dict(color='#000000', width=3))  # line : 외곽테두
리선
)])

fig.show()
```



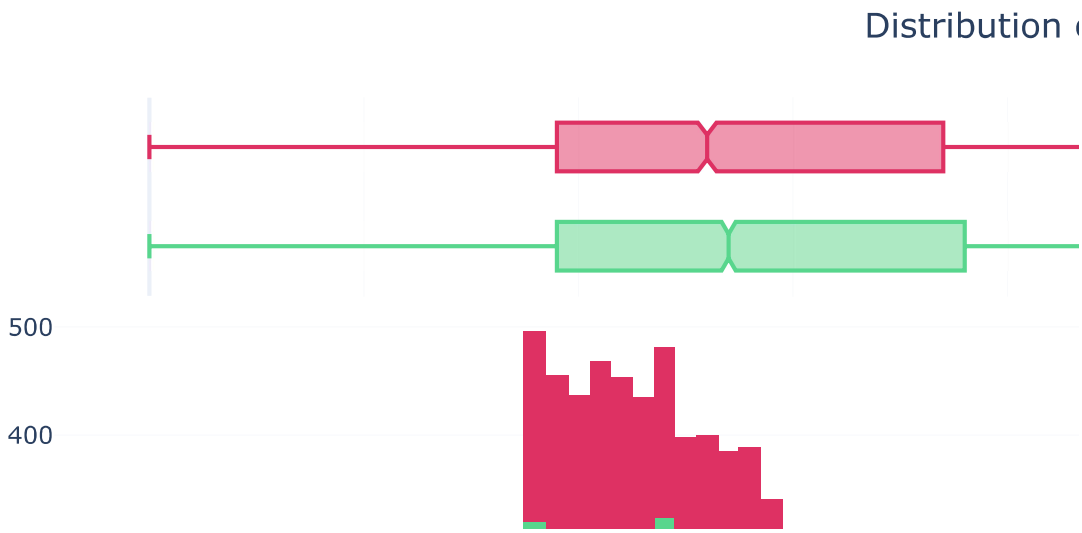
연속형 변수의 피처(Feature) 분포

나이의 분포

In [16]:

```
train_age = train_n.copy()
test_age = test_n.copy()

# 학습용/테스트용 분류를 위한 type컬럼 생성
train_age["type"] = "Train"
test_age["type"] = "Test"

# train, test 결합
ageDf = pd.concat([train_age, test_age])
```

```
# plotly를 활용한 히스토그램 시각화
fig = px.histogram(data_frame = ageDf,
                   x="Age",
                   color= "type",   # train, test의 종류 구분(컬럼)
                   color_discrete_sequence =  ['#58D68D','#DE3163'],
                   marginal="box", # subplot를 추가 가능. (rug, box)
                   nbins= 100,     # 구간 막대 기준 개수
                   template="plotly_white"
                   )

# title_x : 좌우 정렬할 위치(좌:0, 우:1)
fig.update_layout(title = "Distribution of Age" , title_x = 0.5)
fig.show()
```



범주형 변수의 분포

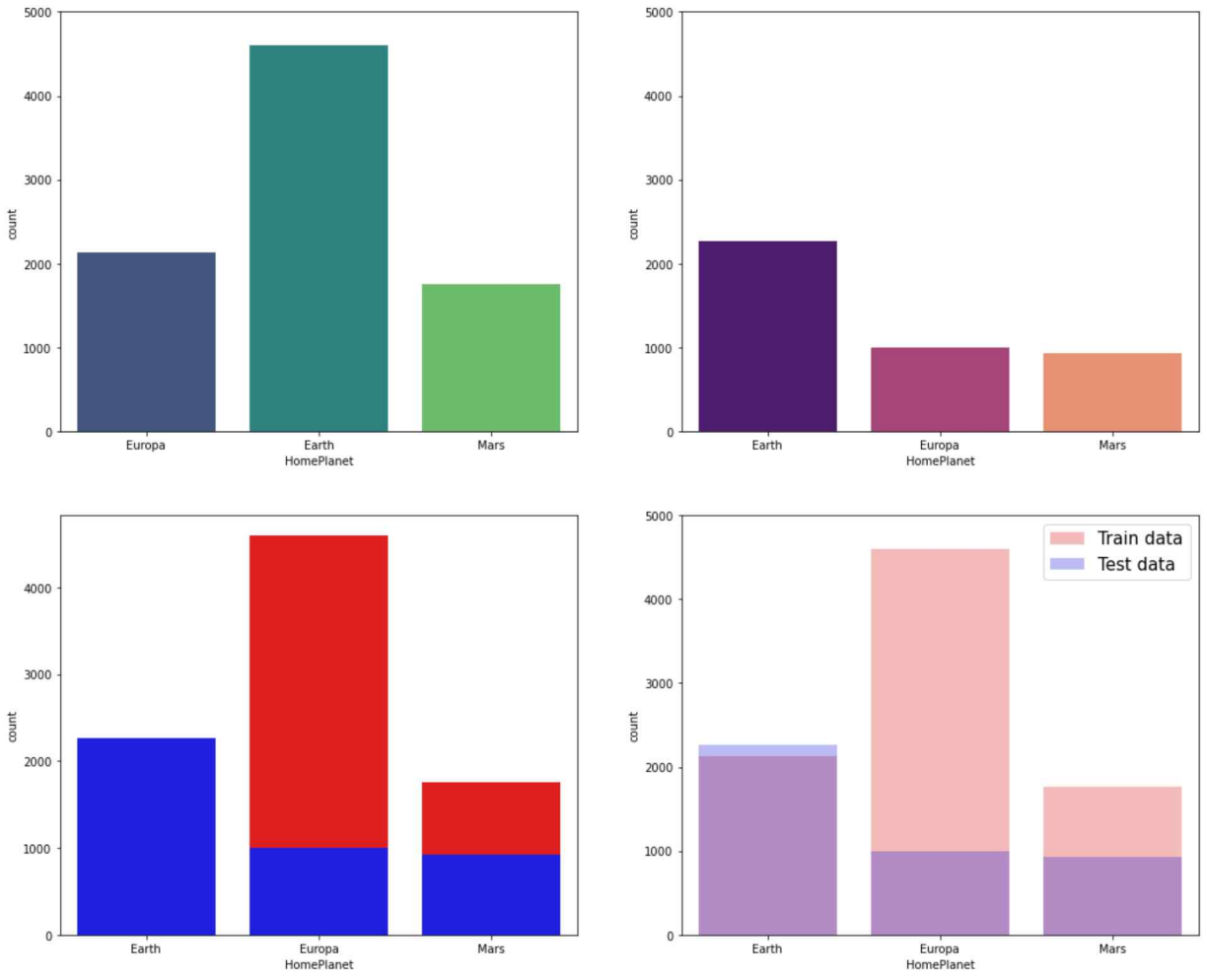
In [17]:

```
fig, axes = plt.subplots(2, 2, figsize=(18, 15))
print( cat_features )

sns.countplot(train_n["HomePlanet"],ax = axes[0,0],palette =
"viridis",label='Train data')
```

```
sns.countplot(test_n["HomePlanet"],ax = axes[0,1],palette =
"magma",label='Test data')

sns.countplot(train_n["HomePlanet"],ax = axes[1,0],
              color = "red",label='Train data')
sns.countplot(test_n["HomePlanet"],ax = axes[1,0],
              color = "blue",label='Test data')

sns.countplot(train_n["HomePlanet"],ax = axes[1,1],
              color = "red",label='Train data', alpha=0.3)
sns.countplot(test_n["HomePlanet"],ax = axes[1,1],
              color = "blue",label='Test data', alpha=0.3)

axes[0,0].set_ylim(0,5000)
axes[0,1].set_ylim(0,5000)
axes[1,1].set_ylim(0,5000)
axes[1,1].legend(fontsize=15)   # 범례
```

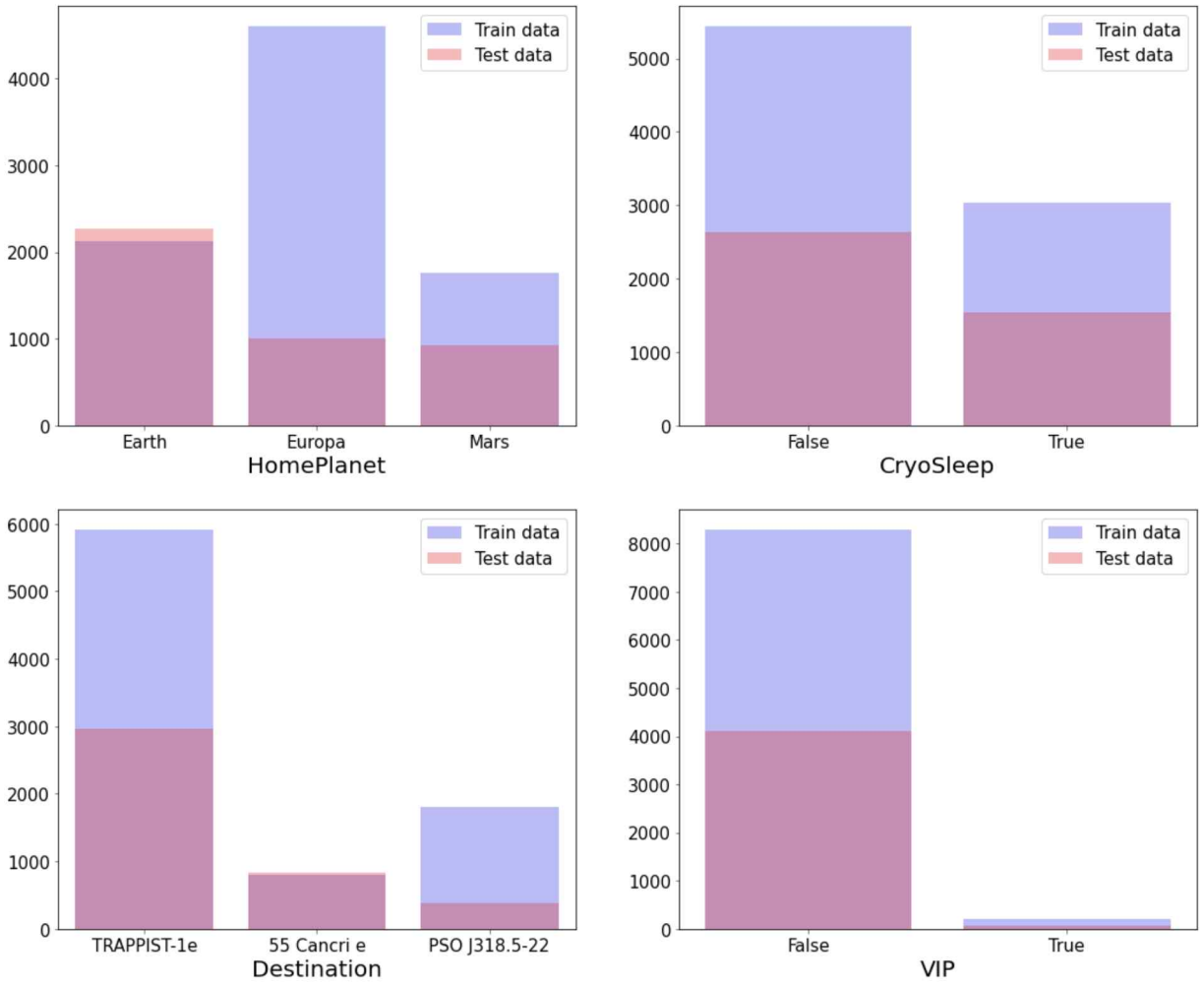
```
['HomePlanet', 'CryoSleep', 'Destination', 'VIP']
```

Out[17]: <matplotlib.legend.Legend at 0x22f2b6f4f70>



In [18]:
```
if len(cat_features) == 0 :
```

```
    print("No Categorical features")
else:
    ncols = 2
    nrows = 2

    fig, axes = plt.subplots(nrows, ncols, figsize=(18, 15))
    for r in range(nrows):
        for c in range(ncols):
            col = cat_features[r*ncols+c]

            sns.countplot(train_n[col],ax = axes[r,c],
                          color="blue",
                          label='Train data', alpha=0.3)
            sns.countplot(test_n[col],ax = axes[r,c],
                          color="red",
                          label='Test data', alpha=0.3)

            axes[r,c].legend(fontsize=15)   # 범례

            # x축, y축 레이블
            axes[r,c].set_ylabel('')
            axes[r,c].set_xlabel(col, fontsize=20)
            axes[r,c].tick_params(labelsize=15, width=0.5)
            axes[r,c].xaxis.offsetText.set_fontsize(4)
            axes[r,c].yaxis.offsetText.set_fontsize(4)
    plt.show()
```



Target 값의 분포

- 두 개의 target 값이 있다.
  - 0과 1
- target값 둘 다 거의 동일한 분포를 갖는다.

In [19]:
```
print( TARGET )
print( train_n[TARGET].value_counts() )


target_df = pd.DataFrame(train_n[TARGET].value_counts()).reset_index()
print(target_df)
```

```
Transported
True      4378
False     4315
Name: Transported, dtype: int64
   index  Transported
0   True         4378
1  False         4315
```

In [20]:
```
target_df = pd.DataFrame(train_n[TARGET].value_counts()).reset_index()
target_df.columns = [TARGET, 'count']


fig = px.bar(data_frame =target_df, x = TARGET, y = 'count'  )
```

```
fig.update_traces(marker_color =['#58D68D','#DE3163'],
                  marker_line_color='rgb(0,0,0)',
                  marker_line_width=2,)

fig.update_layout(title = "Target Distribution",
                  template = "plotly_white",
                  title_x = 0.5)

percent_0 = target_df["count"][0] *100 / train_n.shape[0]
percent_1 = target_df["count"][1]* 100 / train_n.shape[0]

print("Transported(0)의 비율 {:.2f} %".format( percent_0 ))
print("Transported(1)의 비율 {:.2f} %".format( percent_1 ))

fig.show()
```

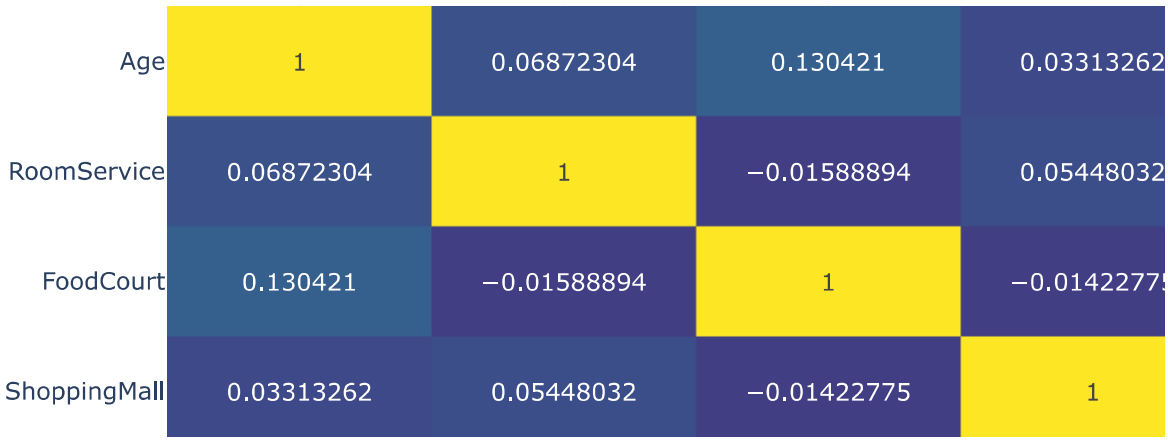
```
Transported(0)의 비율 50.36 %
Transported(1)의 비율 49.64 %
```

상관계수 행렬

In [21]:

```
fig = px.imshow(train_n.corr(),
```

```
                text_auto=True,
                aspect="auto" ,
                color_continuous_scale = "viridis")
fig.show()
```



03. 데이터 전처리

SimpleImputer 예제

In [22]:

```
import numpy as np
import pandas as pd
from sklearn.impute import SimpleImputer

##########데이터 로드

df = pd.DataFrame([
    [2, 1, 3 , 3],
    [3, 2, 4, 5],
    [3, np.nan, 4, 7],
    [5, np.nan, 5, 10],
    [7, 5, 6, 12],
```

```
    [2, 5, np.nan, 7],
    [8, 9, np.nan, 13],
], columns=['hour', 'attendance', 'assignment','score'])



print(df.isnull().sum())
df
```

```
hour          0
attendance    2
assignment    2
score         0
dtype: int64
```

Out[22]:

	hour	attendance	assignment	score
0	2	1.0	3.0	3
1	3	2.0	4.0	5
2	3	NaN	4.0	7
3	5	NaN	5.0	10
4	7	5.0	6.0	12
5	2	5.0	NaN	7
6	8	9.0	NaN	13

In [23]:

```
x_data = df.drop(['score'], axis=1)
y_data = df['score']


transformer = SimpleImputer()
transformer.fit(x_data)
x_data = transformer.transform(x_data)
print(x_data)
```

```
[[2.  1.  3. ]
 [3.  2.  4. ]
 [3.  4.4 4. ]
 [5.  4.4 5. ]
 [7.  5.  6. ]
 [2.  5.  4.4]
 [8.  9.  4.4]]
```

SimpleImputer를 이용한 결측치 처리

In [24]:

```
train_n.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8693 entries, 0 to 8692
Data columns (total 13 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   HomePlanet    8492 non-null   object
 1   CryoSleep     8476 non-null   object
 2   Cabin         8494 non-null   object
 3   Destination   8511 non-null   object
```

```
 4   Age           8514 non-null   float64
 5   VIP           8490 non-null   object
 6   RoomService   8512 non-null   float64
 7   FoodCourt     8510 non-null   float64
 8   ShoppingMall  8485 non-null   float64
 9   Spa           8510 non-null   float64
 10  VRDeck        8505 non-null   float64
 11  Name          8493 non-null   object
 12  Transported   8693 non-null   bool
dtypes: bool(1), float64(6), object(6)
memory usage: 823.6+ KB
```

In [25]:

```
print( train_n["HomePlanet"].value_counts() )
print( test_n["HomePlanet"].value_counts() )
print( train_n["HomePlanet"].unique(), test_n["HomePlanet"].unique() )
```

```
Earth     4602
Europa    2131
Mars      1759
Name: HomePlanet, dtype: int64
Earth     2263
Europa    1002
Mars       925
Name: HomePlanet, dtype: int64
['Europa' 'Earth' 'Mars' nan] ['Earth' 'Europa' 'Mars' nan]
```

In [26]:

```
STRATEGY = 'median'

imputer_cols = ["Age", "RoomService", "FoodCourt",
                "ShoppingMall", "Spa", "VRDeck"]

# 결측치 대치
imputer = SimpleImputer(strategy=STRATEGY )
imputer.fit(train_n[imputer_cols])
train_n[imputer_cols] = imputer.transform(train_n[imputer_cols])
test_n[imputer_cols] = imputer.transform(test_n[imputer_cols])

train_n["HomePlanet"].fillna('Z', inplace=True)
test_n["HomePlanet"].fillna('Z', inplace=True)

print( train_n[imputer_cols].isnull().sum() )
print( test_n[imputer_cols].isnull().sum() )

train_n.isnull().sum(), test_n.isnull().sum()
```

```
Age             0
RoomService     0
FoodCourt       0
ShoppingMall    0
Spa             0
VRDeck          0
dtype: int64
Age             0
RoomService     0
FoodCourt       0
```

```
ShoppingMall    0
Spa             0
VRDeck          0
dtype: int64
```

Out[26]:
```
(HomePlanet       0
 CryoSleep      217
 Cabin          199
 Destination    182
 Age              0
 VIP            203
 RoomService      0
 FoodCourt        0
 ShoppingMall     0
 Spa              0
 VRDeck           0
 Name           200
 Transported      0
 dtype: int64,
 HomePlanet       0
 CryoSleep       93
 Cabin          100
 Destination     92
 Age              0
 VIP             93
 RoomService      0
 FoodCourt        0
 ShoppingMall     0
 Spa              0
 VRDeck           0
 Name            94
 dtype: int64)
```

범주형 변수 인코딩

LabelEncoder()를 이용하여 라벨 인코딩이 가능.

In [27]:
```
label_cols = ["HomePlanet", "CryoSleep","Cabin",
              "Destination" ,"VIP"]


def label_encoder(train,test,columns):
    for col in columns:
        train[col] = train[col].astype(str)
        test[col] = test[col].astype(str)
        train[col] = LabelEncoder().fit_transform(train[col])
        test[col] =  LabelEncoder().fit_transform(test[col])
    return train, test

train_n ,test_n = label_encoder(train_n, test_n ,label_cols)


train_n[label_cols].head()
```

Out[27]:

	HomePlanet	CryoSleep	Cabin	Destination	VIP
0	1	0	149	2	0
1	0	0	2184	2	0
2	1	0	1	2	1

	HomePlanet	CryoSleep	Cabin	Destination	VIP
3	1	0	1	2	0
4	0	0	2186	2	0

In [28]:
```
train_n.drop(["Name" ,"Cabin"] , axis = 1 ,inplace = True)
test_n.drop(["Name" ,"Cabin"] , axis = 1 ,inplace = True)
X = train_n.drop(TARGET , axis =1 )
y = train_n[TARGET]
X_train , X_test , y_train , y_test = train_test_split(X , y,
                                                       random_state = 12
,
                                                       test_size =0.33)
```

04. Modeling

In [29]:
```
train_n.isnull().sum()
```

Out[29]:
```
HomePlanet      0
CryoSleep       0
Destination     0
Age             0
VIP             0
RoomService     0
FoodCourt       0
ShoppingMall    0
Spa             0
VRDeck          0
Transported     0
dtype: int64
```

In [30]:
```
test_n.isnull().sum()
```

Out[30]:
```
HomePlanet      0
CryoSleep       0
Destination     0
Age             0
VIP             0
RoomService     0
FoodCourt       0
ShoppingMall    0
Spa             0
VRDeck          0
dtype: int64
```

In [31]:
```
train_n.head(3)
```

Out[31]:

	HomePlanet	CryoSleep	Destination	Age	VIP	RoomService	FoodCourt	ShoppingMall	Spa
0	1	0	2	39.0	0	0.0	0.0	0.0	0.0
1	0	0	2	24.0	0	109.0	9.0	25.0	549.0
2	1	0	2	58.0	1	43.0	3576.0	0.0	6715.0

In [32]:
```
test_n.head(3)
```

Out[32]:

	HomePlanet	CryoSleep	Destination	Age	VIP	RoomService	FoodCourt	ShoppingMall	Spa
0	0	1	2	27.0	0	0.0	0.0	0.0	0.0
1	0	0	2	19.0	0	0.0	9.0	0.0	2823.0
2	1	1	0	31.0	0	0.0	0.0	0.0	0.0

In [33]:
```
# 기본 모델을 위해 특징은 5개만 선택하는 것으로 한다.
sel = ['HomePlanet', 'CryoSleep', 'Destination', 'Age', 'VIP']

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

last_test = test_n[sel]
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.3,
                                                    random_state = 0)

y_train.shape
```

Out[33]: (6085,)

In [34]:
```
X_train.info(), X_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6085 entries, 7289 to 2732
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   HomePlanet   6085 non-null   int32
 1   CryoSleep    6085 non-null   int32
 2   Destination  6085 non-null   int32
 3   Age          6085 non-null   float64
 4   VIP          6085 non-null   int32
dtypes: float64(1), int32(4)
memory usage: 190.2 KB
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2608 entries, 3601 to 5117
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   HomePlanet   2608 non-null   int32
 1   CryoSleep    2608 non-null   int32
 2   Destination  2608 non-null   int32
 3   Age          2608 non-null   float64
 4   VIP          2608 non-null   int32
dtypes: float64(1), int32(4)
memory usage: 81.5 KB
```

Out[34]: (None, None)

In [35]:
```
from sklearn.neighbors import KNeighborsClassifier
```

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

In [36]:
```
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 :  0.7502054231717338
테스트용 정확도 :  0.6706288343558282
```

In [37]:
```
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 :  0.7391947411668036
테스트용 정확도 :  0.7319785276073619
```

DecisionTreeClassifier 모델이 좋아보임. 우선은 이걸로 최종 모델을 해 본다. **max_depth**값은 최적으로 맞춘다.

In [38]:
```
depth_num = range(1, 7, 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 :  0.7201314708299096
테스트용 정확도 :  0.7120398773006135
max_depth 값 :  2
학습용 정확도 :  0.7377156943303205
테스트용 정확도 :  0.727760736196319
max_depth 값 :  3
학습용 정확도 :  0.7391947411668036
테스트용 정확도 :  0.7319785276073619
max_depth 값 :  4
학습용 정확도 :  0.7403451109285127
테스트용 정확도 :  0.7304447852760736
max_depth 값 :  5
학습용 정확도 :  0.7437962202136401
테스트용 정확도 :  0.7300613496932515
max_depth 값 :  6
학습용 정확도 :  0.7498767460969598
테스트용 정확도 :  0.7292944785276073
```

max_depth=3일때가 가장 좋음. 최종 모델을 의사결정트리 모델로 선택하고 max_depth는 3으로 하자.

In [39]:
```
model1 = DecisionTreeClassifier(max_depth=3, random_state=0)
model1.fit(X_train, y_train)
pred = model1.predict(last_test)
pred[0:10]
```

Out[39]:
```
array([ True, False,  True, False, False, False,  True,  True,  True,
       False])
```

In [40]:
```
### 제출용 파일 생성
sub.columns
```

Out[40]:
```
Index(['PassengerId', 'Transported'], dtype='object')
```

In [41]:
```
sub['Transported'] = pred
sub.to_csv("./data/Space_Titanic/first_sub.csv", index=False)
```

베이스라인 모델 점수 : **Score: 0.74374**