실전 데이터 분석 - Spaceship Titanic 데이터 셋

- pycaret를 활용한 automl 실습해 보기

학습 내용

- 1. kaggle 데이터 셋 가져오기
- 2. 라이브러리 및 데이터 셋 가져오기
- 3. 데이터 EDA
- 4. 데이터 전처리
- 5. 모델 구축 및 예측

목차

01. kaggle 데이터 셋 가져오기

목차로 이동하기

대회 개요

- 대회 : https://www.kaggle.com/competitions/spaceship-titanic (https://www.kaggle.com/competitions/spaceship-titanic)
- 참조 노트북 : https://www.kaggle.com/code/arootda/pycaret-visualization-optimization-0-81 (https://www.kaggle.com/code/arootda/pycaret-visualization-optimization-0-81)

사전 준비

- 개발 환경 : google colab CPU (같은 코드로 GPU 가능 - 2022/07)
- kaggle 설치
- kaggle의 API Token 다운로드
  - 본인 Account페이지 -
  - API - Create New API Token 선택 후, 파일 다운로드
  - 파일(kaggle.json)을 코랩에 업로드
- 대회 : https://www.kaggle.com/competitions/spaceship-titanic/data (https://www.kaggle.com/competitions/spaceship-titanic/data)
- data를 선택 후, 다운로드 cmd 확인
- 대회 참여을 선택 후, 'I Understand and Accept'가 먼저 수행되어야 한다.

구글 드라이브에서 kaggle.json 가져오기 - 방법1

- 구글 드라이브를 연결 후, Colab으로 복사해서 가져오기

```
from google.colab import drive
drive.mount('/content/drive')
```

설치

```
!pip install kaggle --upgrade
```

구글 드라이브에서 kaggle.json 가져오기 - 방법2

kaggle.json 파일 이동

```
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
!ls -l kaggle.json
```

필요한 데이터 셋 가져오기

```
!kaggle competitions download -c spaceship-titanic
```

데이터 확인

```
# 파일 확인
!ls -l

# 압축 풀기
!unzip [파일명]

# 파일 확인
!ls -l
```

In [1]:

```
# 구글 드라이브 연결
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [2]:

```
# kaggle 패키지 설치
!pip install kaggle --upgrade
```

```
Looking in indexes: https://pypi.org/simple, (https://pypi.org/simpl
e,) https://us-python.pkg.dev/colab-wheels/public/simple/ (https://us-
python.pkg.dev/colab-wheels/public/simple/)
Requirement already satisfied: kaggle in /usr/local/lib/python3.7/dist
-packages (1.5.12)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.7/dis
t-packages (from kaggle) (1.24.3)
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dis
t-packages (from kaggle) (2022.6.15)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.7/d
ist-packages (from kaggle) (1.15.0)
Requirement already satisfied: requests in /usr/local/lib/python3.7/di
st-packages (from kaggle) (2.23.0)
Requirement already satisfied: python-slugify in /usr/local/lib/python
3.7/dist-packages (from kaggle) (6.1.2)
Requirement already satisfied: python-dateutil in /usr/local/lib/pytho
n3.7/dist-packages (from kaggle) (2.8.2)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-p
ackages (from kaggle) (4.64.0)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/p
ython3.7/dist-packages (from python-slugify->kaggle) (1.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.
7/dist-packages (from requests->kaggle) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/pyt
hon3.7/dist-packages (from requests->kaggle) (3.0.4)
```

In [3]:

```
### 방법1 - 구글 드라이브에서 kaggle.json을 가져오기
!ls -l drive/MyDrive/
!cp drive/MyDrive/kaggle.json /content/
!ls -l /content/
```

```
total 902
drwx------ 2 root root   4096 Aug  8  2020  탈잉
drwx------ 2 root root   4096 Sep  9  2020  유튜브
drwx------ 2 root root   4096 Sep  1  2020  방송통신대학교_파이썬기본
-rw------- 1 root root  18644 Sep 25  2020  1_7_내장함수.ipynb
drwx------ 2 root root   4096 Aug  3  2020  책번역_20200803
drwx------ 2 root root   4096 May  8  2020  빅데이터4기_서울IT
drwx------ 2 root root   4096 Sep 20  2020  AI이노베이션
drwx------ 2 root root   4096 Jun 17  2020  Burj_Khalifa
drwx------ 2 root root   4096 Jan  3  2020 'Colab Notebooks'
drwx------ 2 root root   4096 Jun 17  2020  Colosseum
drwx------ 2 root root   4096 Jul 20  2020  dataset
-rw------- 1 root root  10842 Nov 18  2020  gan_deep_dream.ipynb
drwx------ 2 root root   4096 May  1  2021  jds
-rw------- 1 root root 398659 Mar 24  2021  샘플이미지.jpg
-rw------- 1 root root     67 Jun 28 14:57  kaggle.json
-rw------- 1 root root 444540 Jan 26  2021  서약.png
drwx------ 2 root root   4096 Jun 17  2020  test
total 12
drwx------ 5 root root 4096 Jun 29 09:23 drive
-rw------- 1 root root   67 Jun 29 09:23 kaggle.json
drwxr-xr-x 1 root root 4096 Jun 15 13:42 sample_data
```

In [4]:

```
# kaggle.json 업로드 후,
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
!ls -l kaggle.json
```

```
-rw------- 1 root root 67 Jun 29 09:23 kaggle.json
```

In [5]:

```
# kaggle API를 이용하여 데이터 셋 다운로드 하기
!kaggle competitions download -c spaceship-titanic
```

```
Downloading spaceship-titanic.zip to /content
  0% 0.00/299k [00:00<?, ?B/s]
100% 299k/299k [00:00<00:00, 83.1MB/s]
```

In [6]:

```
# 파일 확인
!ls -l

# 압축 풀기
!unzip spaceship-titanic.zip

!ls -l
```

```
total 312
drwx------ 5 root root   4096 Jun 29 09:23 drive
-rw------- 1 root root     67 Jun 29 09:23 kaggle.json
drwxr-xr-x 1 root root   4096 Jun 15 13:42 sample_data
-rw-r--r-- 1 root root 306403 Jun 29 09:23 spaceship-titanic.zip
Archive:  spaceship-titanic.zip
  inflating: sample_submission.csv
  inflating: test.csv
  inflating: train.csv
total 1524
drwx------ 5 root root   4096 Jun 29 09:23 drive
-rw------- 1 root root     67 Jun 29 09:23 kaggle.json
drwxr-xr-x 1 root root   4096 Jun 15 13:42 sample_data
-rw-r--r-- 1 root root  59902 Feb 11 14:02 sample_submission.csv
-rw-r--r-- 1 root root 306403 Jun 29 09:23 spaceship-titanic.zip
-rw-r--r-- 1 root root 372487 Feb 11 14:02 test.csv
-rw-r--r-- 1 root root 805421 Feb 11 14:02 train.csv
```

02. 라이브러리 준비 및 데이터 셋 가져오기

목차로 이동하기

In [7]:

```
!pip install pycaret==2.3.10
```

```
Looking in indexes: https://pypi.org/simple, (https://pypi.org/simp
le,) https://us-python.pkg.dev/colab-wheels/public/simple/ (http
s://us-python.pkg.dev/colab-wheels/public/simple/)
Collecting pycaret==2.3.10
  Downloading pycaret-2.3.10-py3-none-any.whl (320 kB)
     |████████████████████████████████| 320 kB 4.2 M
B/s
Requirement already satisfied: IPython in /usr/local/lib/python3.7/
dist-packages (from pycaret==2.3.10) (5.5.0)
Collecting umap-learn
  Downloading umap-learn-0.5.3.tar.gz (88 kB)
     |████████████████████████████████| 88 kB 8.5 MB/
s
Collecting mlflow
  Downloading mlflow-1.27.0-py3-none-any.whl (17.9 MB)
     |████████████████████████████████| 17.9 MB 576 k
B/s
Collecting mlxtend>=0.17.0
  Downloading mlxtend-0.20.0-py2.py3-none-any.whl (1.3 MB)
     |████████████████████████████████| 1.3 MB 40.4 M
```

- 몇몇 패키지가 버전이 맞지 않아, ERROR이 발생하지만, 아래 코드 실행이 가능함.

In [2]:

```
import pycaret
from pycaret.classification import *
from IPython.display import Image, display
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
---------------------------------------------------------------------
--------
ImportError                               Traceback (most recent ca
ll last)
<ipython-input-2-be86b9534894> in <module>()
      1 import pycaret
----> 2 from pycaret.classification import *
      3 from IPython.display import Image, display
      4 import pandas as pd
      5 import matplotlib.pyplot as plt

/usr/local/lib/python3.7/dist-packages/pycaret/classification.py in
<module>()
      8 import numpy as np
      9
---> 10 import pycaret.internal.tabular
     11 from pycaret.loggers.base_logger import BaseLogger
     12 from pycaret.parallel import ParallelBackend

/usr/local/lib/python3.7/dist-packages/pycaret/internal/tabular.py
 in <module>()
     14     get_estimator_from_meta_estimator,
     15 )
---> 16 from pycaret.internal.pipeline import (
     17     add_estimator_to_pipeline,
     18     get_pipeline_estimator_label,

/usr/local/lib/python3.7/dist-packages/pycaret/internal/pipeline.py
in <module>()
      9 # This pipeline is only to be used internally.
     10
---> 11 from pycaret.internal.utils import get_all_object_vars_and_
properties, is_fit_var
     12 import imblearn.pipeline
     13 from sklearn.utils import _print_elapsed_time

/usr/local/lib/python3.7/dist-packages/pycaret/internal/utils.py in
<module>()
      8 from pycaret.containers.models.base_model import ModelConta
iner
      9 import pandas as pd
---> 10 import pandas.io.formats.style
     11 import ipywidgets as ipw
     12 from IPython.display import display, HTML, clear_output, up
date_display

/usr/local/lib/python3.7/dist-packages/pandas/io/formats/style.py i
n <module>()
     47 from pandas.io.formats.format import save_to_buffer
     48
---> 49 jinja2 = import_optional_dependency("jinja2", extra="DataFr
```

```
ame.style requires jinja2.")
     50
     51 from pandas.io.formats.style_render import (

/usr/local/lib/python3.7/dist-packages/pandas/compat/_optional.py i
n import_optional_dependency(name, extra, errors, min_version)
    116     except ImportError:
    117         if errors == "raise":
--> 118             raise ImportError(msg) from None
    119         else:
    120             return None

ImportError: Missing optional dependency 'Jinja2'. DataFrame.style
 requires jinja2. Use pip or conda to install Jinja2.

--------------------------------------------------------------------
--------
NOTE: If your import is failing due to a missing package, you can
manually install dependencies using either !pip or !apt.

To view examples of installing some common dependencies, click the
"Open Examples" button below.
--------------------------------------------------------------------
--------
```

- 에러 발생시

  ```
  from pycaret.classification import *

  ImportError: Missing optional dependency 'Jinja2'. DataFrame.style requires
   jinja2. Use pip or conda to install Jinja2.
  ```

해결 시도

- 관련 패키지를 설치

  ```
  pip3 install jinja2==3.0.1
  pip install markupsafe==2.0.1
  ```

In [3]:

```
!pip install jinja2==3.0.1
!pip install markupsafe==2.0.1
```

```
Looking in indexes: https://pypi.org/simple, (https://pypi.org/simpl
e,) https://us-python.pkg.dev/colab-wheels/public/simple/ (https://us-
python.pkg.dev/colab-wheels/public/simple/)
Collecting jinja2==3.0.1
  Downloading Jinja2-3.0.1-py3-none-any.whl (133 kB)
     |████████████████████████████████████████| 133 kB 4.1 MB/s
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/pytho
n3.7/dist-packages (from jinja2==3.0.1) (2.1.1)
Installing collected packages: jinja2
  Attempting uninstall: jinja2
    Found existing installation: Jinja2 2.11.3
    Uninstalling Jinja2-2.11.3:
      Successfully uninstalled Jinja2-2.11.3
ERROR: pip's dependency resolver does not currently take into account
 all the packages that are installed. This behaviour is the source of
 the following dependency conflicts.
google-colab 1.0.0 requires requests~=2.23.0, but you have requests 2.
28.0 which is incompatible.
flask 1.1.4 requires Jinja2<3.0,>=2.10.1, but you have jinja2 3.0.1 wh
ich is incompatible.
datascience 0.10.6 requires folium==0.2.1, but you have folium 0.8.3 w
hich is incompatible.
Successfully installed jinja2-3.0.1
Looking in indexes: https://pypi.org/simple, (https://pypi.org/simpl
e,) https://us-python.pkg.dev/colab-wheels/public/simple/ (https://us-
python.pkg.dev/colab-wheels/public/simple/)
Collecting markupsafe==2.0.1
  Downloading MarkupSafe-2.0.1-cp37-cp37m-manylinux_2_5_x86_64.manylin
ux1_x86_64.manylinux_2_12_x86_64.manylinux2010_x86_64.whl (31 kB)
Installing collected packages: markupsafe
  Attempting uninstall: markupsafe
    Found existing installation: MarkupSafe 2.1.1
    Uninstalling MarkupSafe-2.1.1:
      Successfully uninstalled MarkupSafe-2.1.1
ERROR: pip's dependency resolver does not currently take into account
 all the packages that are installed. This behaviour is the source of
 the following dependency conflicts.
pandas-profiling 3.2.0 requires markupsafe~=2.1.1, but you have markup
safe 2.0.1 which is incompatible.
flask 1.1.4 requires Jinja2<3.0,>=2.10.1, but you have jinja2 3.0.1 wh
ich is incompatible.
datascience 0.10.6 requires folium==0.2.1, but you have folium 0.8.3 w
hich is incompatible.
Successfully installed markupsafe-2.0.1
```

적용을 위해 재기동 후, 실행

In [1]:

```
# 메모리 정리
# gc.collect() : 가비지 컬렉션을 수행
import gc
gc.collect()
```

Out[1]:

```
128
```

In [2]:

```
import pycaret
from pycaret.classification import *
from IPython.display import Image, display
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

print(pycaret.__version__)
print(sns.__version__)
print(pd.__version__)
```

```
/usr/local/lib/python3.7/dist-packages/distributed/config.py:20: YAMLL
oadWarning: calling yaml.load() without Loader=... is deprecated, as t
he default Loader is unsafe. Please read https://msg.pyyaml.org/load
 (https://msg.pyyaml.org/load) for full details.
  defaults = yaml.load(f)

2.3.10
0.11.2
1.3.5
```

In [3]:

```
# 기본 라이브러리
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

# 머신러닝 관련 라이브러리
from sklearn.impute import SimpleImputer
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import StratifiedKFold, train_test_split

# 시각화 관련 라이브러리
import plotly.express as px
from plotly.subplots import make_subplots
import plotly.graph_objects as go

PALETTE=['lightcoral', 'lightskyblue', 'gold', 'sandybrown', 'navajowhite',
        'khaki', 'lightslategrey', 'turquoise', 'rosybrown', 'thistle', 'pink']

sns.set_palette(PALETTE)
BACKCOLOR = '#f6f5f5'

from IPython.core.display import HTML
```

In [4]:

```
def multi_table(table_list):
    return HTML(
        f"<table><tr> {''.join(['<td>' + table._repr_html_() + '</td>' for table in
```

In [5]:

```
# 데이터 불러오기
train = pd.read_csv("train.csv")
test = pd.read_csv("test.csv")
submission = pd.read_csv("sample_submission.csv")

all_data = pd.concat([train, test], axis=0)
```

03. 데이터 EDA

목차로 이동하기

In [6]:

```
all_data.head(10).style.background_gradient()
```

Out[6]:

	PassengerId	HomePlanet	CryoSleep	Cabin	Destination	Age	VIP	RoomService	F
0	0001_01	Europa	False	B/0/P	TRAPPIST-1e	39.000000	False	0.000000	
1	0002_01	Earth	False	F/0/S	TRAPPIST-1e	24.000000	False	109.000000	
2	0003_01	Europa	False	A/0/S	TRAPPIST-1e	58.000000	True	43.000000	35
3	0003_02	Europa	False	A/0/S	TRAPPIST-1e	33.000000	False	0.000000	12
4	0004_01	Earth	False	F/1/S	TRAPPIST-1e	16.000000	False	303.000000	
5	0005_01	Earth	False	F/0/P	PSO J318.5-22	44.000000	False	0.000000	4
6	0006_01	Earth	False	F/2/S	TRAPPIST-1e	26.000000	False	42.000000	15
7	0006_02	Earth	True	G/0/S	TRAPPIST-1e	28.000000	False	0.000000	
8	0007_01	Earth	False	F/3/S	TRAPPIST-1e	35.000000	False	0.000000	7
9	0008_01	Europa	True	B/1/P	55 Cancri e	14.000000	False	0.000000	

In [7]:

```
print(f'train size : {train.shape[0]} x {train.shape[1]}')
print(f'test size : {test.shape[0]} x {test.shape[1]}')
print(f'total size : {all_data.shape[0]} x {all_data.shape[1]}')
```

```
train size : 8693 x 14
test size : 4277 x 13
total size : 12970 x 14
```

In [8]:

```
display(all_data.columns)
print()
display(all_data.dtypes)
print()
display(all_data.info())
```

```
Index(['PassengerId', 'HomePlanet', 'CryoSleep', 'Cabin', 'Destinatio
n', 'Age',
       'VIP', 'RoomService', 'FoodCourt', 'ShoppingMall', 'Spa', 'VRDe
ck',
       'Name', 'Transported'],
      dtype='object')



PassengerId      object
HomePlanet       object
CryoSleep        object
Cabin            object
Destination      object
Age             float64
VIP              object
RoomService     float64
FoodCourt       float64
ShoppingMall    float64
Spa             float64
VRDeck          float64
Name             object
Transported      object
dtype: object


<class 'pandas.core.frame.DataFrame'>
Int64Index: 12970 entries, 0 to 4276
Data columns (total 14 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   PassengerId   12970 non-null  object
 1   HomePlanet    12682 non-null  object
 2   CryoSleep     12660 non-null  object
 3   Cabin         12671 non-null  object
 4   Destination   12696 non-null  object
 5   Age           12700 non-null  float64
 6   VIP           12674 non-null  object
 7   RoomService   12707 non-null  float64
 8   FoodCourt     12681 non-null  float64
 9   ShoppingMall  12664 non-null  float64
 10  Spa           12686 non-null  float64
 11  VRDeck        12702 non-null  float64
 12  Name          12676 non-null  object
 13  Transported   8693 non-null   object
dtypes: float64(6), object(8)
memory usage: 1.5+ MB

None
```

In [9]:

```
nominal_vars = ['HomePlanet', 'CryoSleep', 'Cabin', 'Desination', 'VIP', 'Name']
continuous_vars = ['Age', 'RoomService', 'FoodCourt', 'ShoppingMall', 'Spa', 'VRDeck
target = 'Transported'
```

In [10]:

```
train_st = train[continuous_vars].describe()
test_st = test[continuous_vars].describe()
all_st = all_data[continuous_vars].describe()

multi_table([all_st, train_st, test_st])
```

Out[10]:

	Age	RoomService	FoodCourt	ShoppingMall	Spa	VRDeck
count	12700.000000	12707.000000	12681.000000	12664.000000	12686.000000	12702.000000
mean	28.771969	222.897852	451.961675	174.906033	308.476904	306.789482
std	14.387261	647.596664	1584.370747	590.558690	1130.279641	1180.097223
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	49.000000	77.000000	29.000000	57.000000	42.000000
max	79.000000	14327.000000	29813.000000	23492.000000	22408.000000	24133.000000

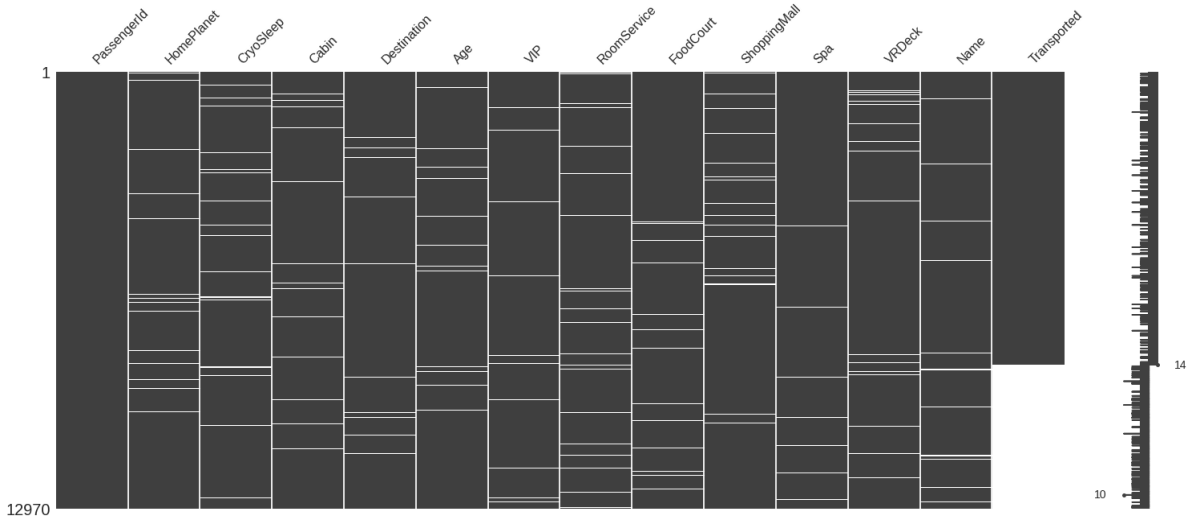
결측치 확인

In [11]:

```
import missingno as msno
msno.matrix(all_data)
```

Out[11]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8411feed10>
```

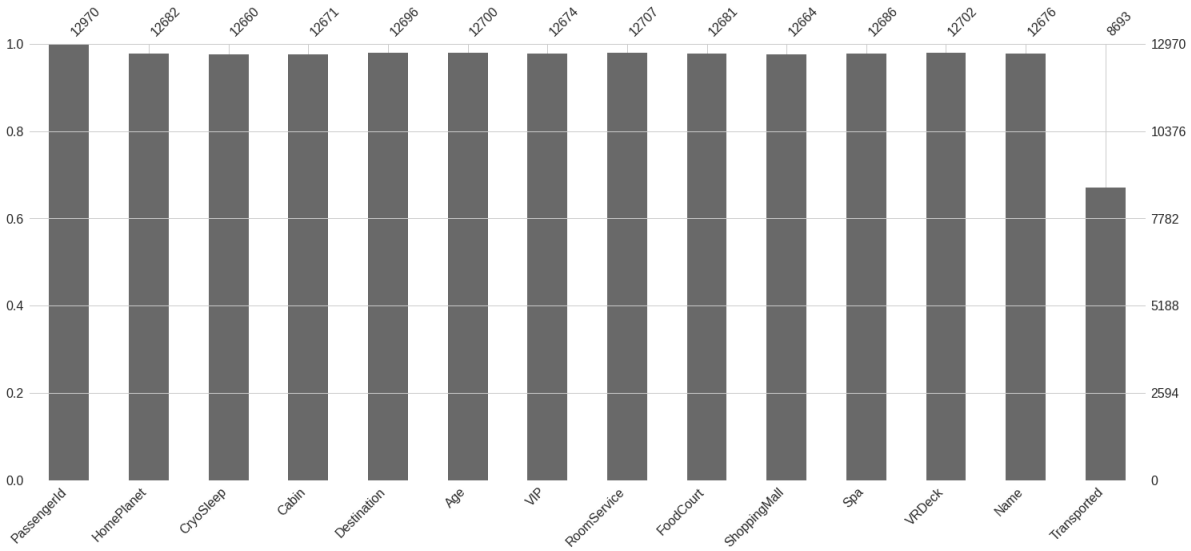


In [12]:

```
msno.bar(all_data)
```

Out[12]:

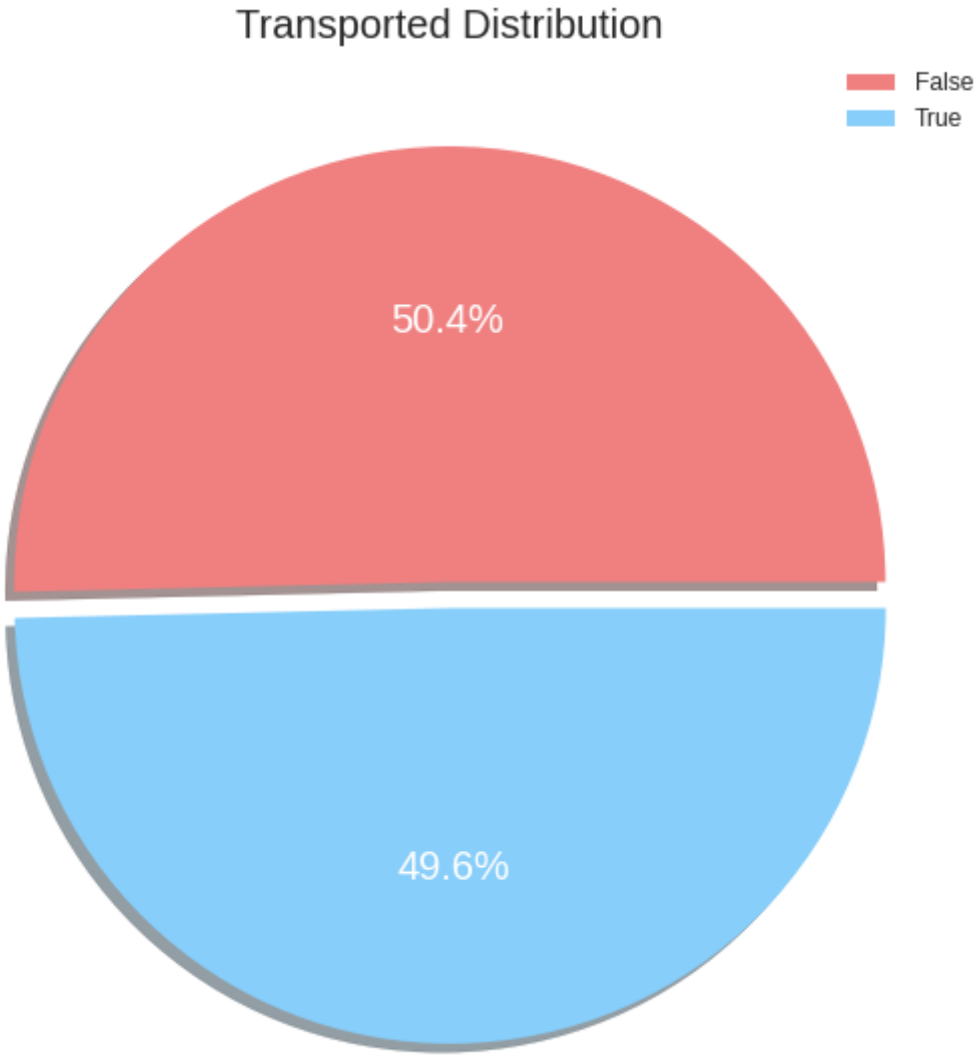
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8411f54690>
```



In [13]:

```
# Target 컬럼(Transported) 시각화

plt.subplots(figsize=(25, 10))
plt.pie(train.Transported.value_counts(),
        shadow=True, explode=[.03,.03],
        autopct='%1.1f%%',
        textprops={'fontsize': 20, 'color': 'white'})
plt.title('Transported Distribution', size=20)
plt.legend(['False', 'True'], loc='best', fontsize=12)
plt.show()
```

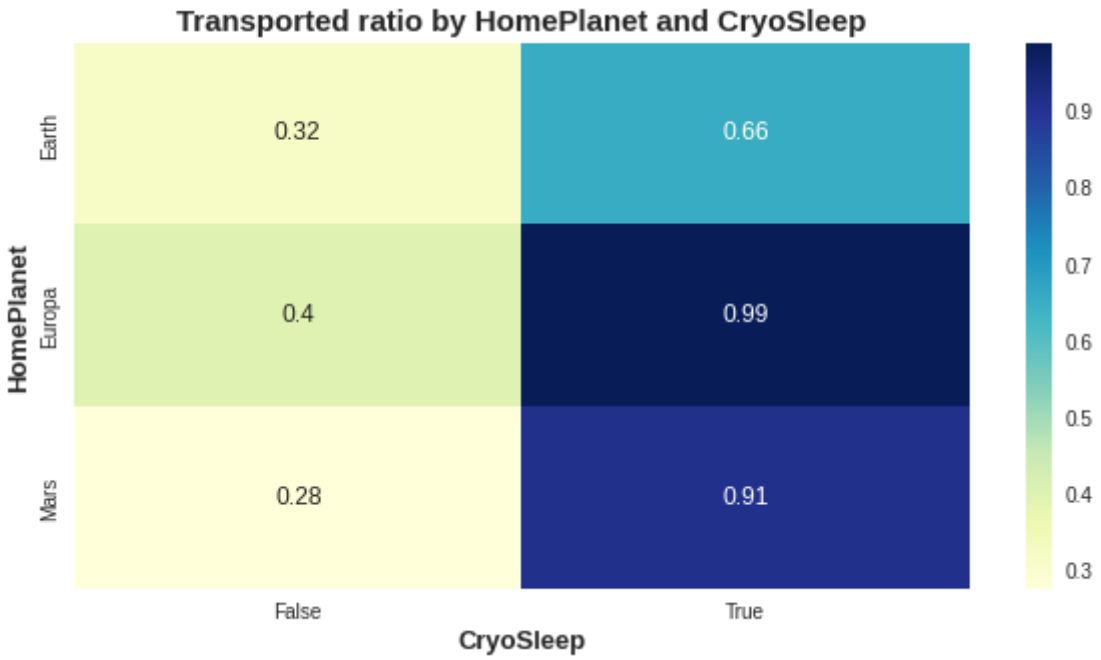


In [14]:

```
# 히트맵을 이용한 시각화
# 데이터의 비율을 표시
# Heatmap can visualize continuous values (or binary variables) in categories and ca
plt.subplots(figsize=(10, 5))
g = sns.heatmap(train.pivot_table(index='HomePlanet',
                                  columns='CryoSleep',
                                  values='Transported'), annot=True, cmap="YlGnBu")
# 제목, x,y레이블
g.set_title('Transported ratio by HomePlanet and CryoSleep',
            weight='bold', size=15)
g.set_xlabel('CryoSleep',
             weight='bold', size=13)

g.set_ylabel('HomePlanet',
             weight='bold', size=13)
plt.show()


pd.crosstab([train.CryoSleep,
             train.Transported],
            train.HomePlanet,margins=True).style.background_gradient()
```



Out[14]:

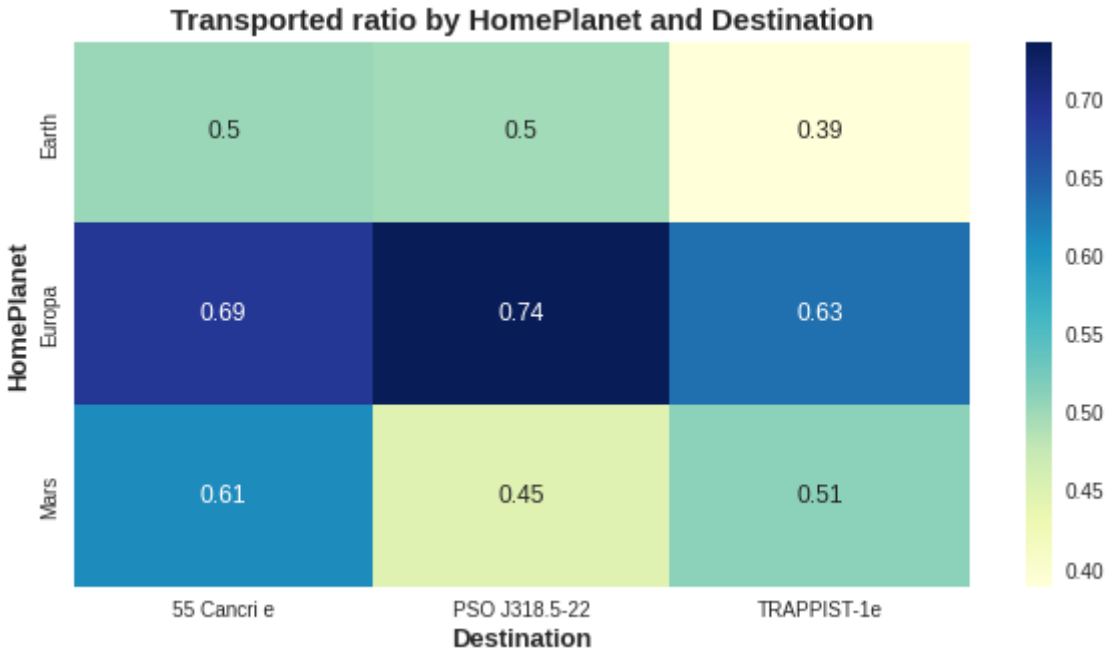
CryoSleep	Transported	HomePlanet	Earth	Europa	Mars	All
False	False		2109	697	757	3563
	True		997	465	290	1752
True	False		475	10	59	544
	True		907	901	610	2418

HomePlanet	Earth	Europa	Mars	All
CryoSleep Transported				
All	4488	2073	1716	8277

In [15]:

```
# Target의 데이터의 전체에서 True가 가지는 비율 시각화
# y축 : HomePlanet, x축 : Destination, 값 : Transported
plt.subplots(figsize=(10, 5))
g = sns.heatmap(train.pivot_table(index='HomePlanet',
                                  columns='Destination',
                                  values='Transported'),
                annot=True, cmap="YlGnBu")
g.set_title('Transported ratio by HomePlanet and Destination',
            weight='bold', size=15)
g.set_xlabel('Destination',
             weight='bold', size=13)
g.set_ylabel('HomePlanet',
             weight='bold', size=13)
plt.show()

pd.crosstab([train.Destination, train.Transported],
            train.HomePlanet,margins=True).style.background_gradient()
```



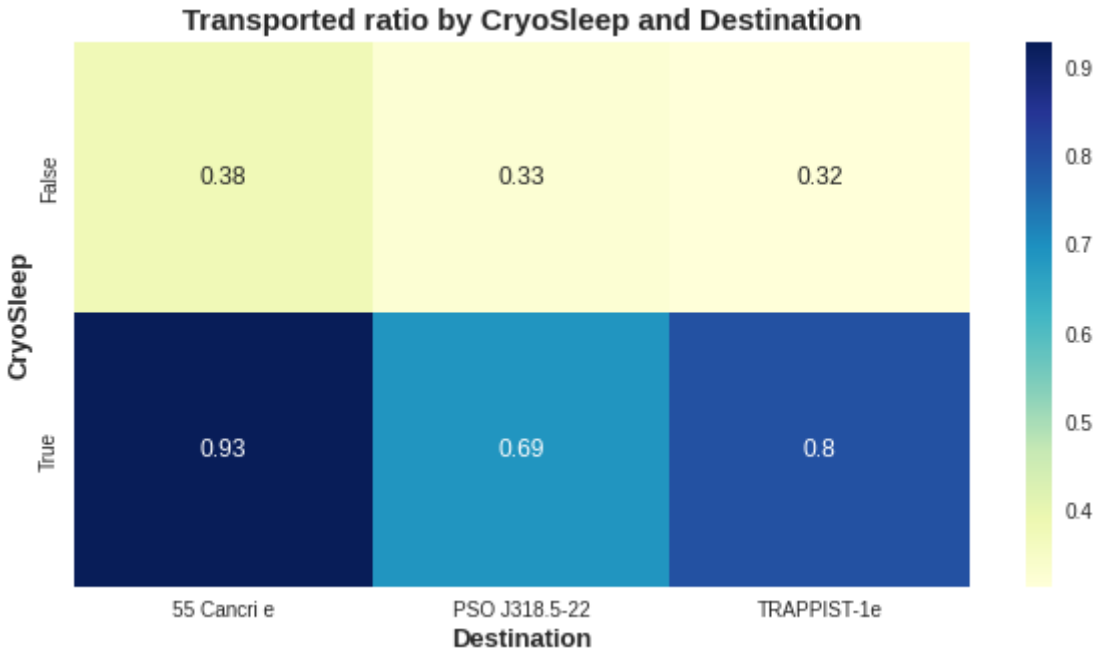
Out[15]:

Destination	Transported	HomePlanet Earth	Europa	Mars	All
55 Cancri e	False	342	275	75	692
	True	348	611	118	1077
PSO J318.5-22	False	357	5	27	389
	True	355	14	22	391
TRAPPIST-1e	False	1894	434	720	3048
	True	1207	755	755	2717
All		4503	2094	1717	8314

In [16]:

```
plt.subplots(figsize=(10, 5))
g = sns.heatmap(train.pivot_table(index='CryoSleep',
                                  columns='Destination',
                                  values='Transported'),
                annot=True, cmap="YlGnBu")
g.set_title('Transported ratio by CryoSleep and Destination',
            weight='bold', size=15)
g.set_xlabel('Destination',
             weight='bold', size=13)
g.set_ylabel('CryoSleep',
             weight='bold', size=13)
plt.show()

pd.crosstab([train.CryoSleep, train.Transported],
            train.Destination,margins=True).style.background_gradient()
```



Out[16]:

	Destination	55 Cancri e	PSO J318.5-22	TRAPPIST-1e	All
CryoSleep	Transported				
False	False	630	265	2669	3564
	True	387	129	1229	1745
True	False	53	119	379	551
	True	686	264	1488	2438
All		1756	777	5765	8298

데이터 전처리

결측치 채우기

In [18]:

```
# CryoSleep의 값 빈도 확인
print( all_data['CryoSleep'].value_counts() )  # 다수의 값이 Fasle를 가진다.
```

```
False    8079
True     4581
Name: CryoSleep, dtype: int64
```

In [19]:

```
# 결측치 처리. 범주의 형일 경우, 다수의 값으로 채운다.
# Replace categorical variables with specific values (False, None) or freeest values
all_data['CryoSleep'].fillna(False, inplace=True)
all_data['Cabin'].fillna('None', inplace=True)
all_data['VIP'].fillna(all_data.VIP.mode()[0], inplace=True)
all_data['HomePlanet'].fillna(all_data.HomePlanet.mode()[0], inplace=True)
all_data['Destination'].fillna(all_data.Destination.mode()[0], inplace=True)

# 연속형 변수의 결측치는 0 또는 평균으로 대치시킨다.
all_data['Age'].fillna(all_data.Age.mean(), inplace=True)
all_data[['RoomService', 'FoodCourt', 'ShoppingMall', 'Spa', 'VRDeck']] =\
                all_data[['RoomService', 'FoodCourt', 'ShoppingMall', 'Spa', 'VRDeck
```

In [20]:

```
# As mentioned earlier, create a new variable by decomposing strings in Cabin and Pa
all_data['Deck'] = all_data.Cabin.apply(lambda x:str(x)[:1])
all_data['Side'] = all_data.Cabin.apply(lambda x:str(x)[-1:])
all_data['PassengerGroup'] = all_data['PassengerId'].apply(lambda x: x.split('_')[0]
all_data['PassengerNo'] = all_data['PassengerId'].apply(lambda x: x.split('_')[1])

# 컬럼에서의 다양한 서비스와 관련 있는 내용을 이용하여 새로운 변수를 생성.
all_data['TotalSpend'] = all_data['RoomService'] + all_data['FoodCourt'] +\
                            all_data['ShoppingMall'] + all_data['Spa'] + all_data[
all_data['PctRoomService'] = all_data['RoomService']/all_data['TotalSpend']
all_data['PctFoodCourt'] = all_data['FoodCourt']/all_data['TotalSpend']
all_data['PctShoppingMall'] = all_data['ShoppingMall']/all_data['TotalSpend']
all_data['PctSpa'] = all_data['Spa']/all_data['TotalSpend']
all_data['PctVRDeck'] = all_data['VRDeck']/all_data['TotalSpend']

# Create new variables by dividing age groups.
all_data['AgeBin'] = 7
for i in range(6):
    all_data.loc[(all_data.Age >= 10*i) & (all_data.Age < 10*(i + 1)), 'AgeBin'] = i
```

In [21]:

```
# 생성된 변수의 결측치가 있다면 이를 0으로 결측치 처리
fill_cols = ['PctRoomService', 'PctFoodCourt', 'PctShoppingMall', 'PctSpa', 'PctVRDe
all_data[fill_cols] = all_data[fill_cols].fillna(0)
```

변수 제거

In [22]:

```
all_data.drop(['PassengerId', 'Name', 'Cabin'], axis=1, inplace=True)
```

라벨 인코딩

In [23]:

```
for col in all_data.columns[all_data.dtypes == object]:
    if col != 'Transported':
        le = LabelEncoder()
        all_data[col] = le.fit_transform(all_data[col])

all_data['CryoSleep'] = all_data['CryoSleep'].astype('int')
all_data['VIP'] = all_data['VIP'].astype('int')
```

데이터 나누기

In [24]:

```
train, X_test = all_data.iloc[ :train.shape[0]], all_data.iloc[train.shape[0]:].drop
X_train, y_train = train.drop(['Transported'], axis=1), train['Transported']
```

05. 모델 구축 및 예측

목차로 이동하기

- pycaret를 활용.

In [25]:

```
s = setup(data=train,
          session_id=7010,
          target='Transported',
          train_size=0.99,
          fold_strategy='stratifiedkfold',
          fold=5,
          fold_shuffle=True,
          silent=True,
          ignore_low_variance=True,
          remove_multicollinearity = True,
          normalize = True,
          normalize_method = 'robust',)
```

	Description	Value
0	session_id	7010
1	Target	Transported
2	Target Type	Binary
3	Label Encoded	False: 0, True: 1
4	Original Data	(8693, 22)
5	Missing Values	False
6	Numeric Features	13
7	Categorical Features	8
8	Ordinal Features	False
9	High Cardinality Features	False
10	High Cardinality Method	None
11	Transformed Train Set	(8606, 45)
12	Transformed Test Set	(87, 45)
13	Shuffle Train-Test	True
14	Stratify Train-Test	False
15	Fold Generator	StratifiedKFold
16	Fold Number	5
17	CPU Jobs	-1
18	Use GPU	False
19	Log Experiment	False
20	Experiment Name	clf-default-name
21	USI	3767
22	Imputation Type	simple
23	Iterative Imputation Iteration	None
24	Numeric Imputer	mean
25	Iterative Imputation Numeric Model	None
26	Categorical Imputer	constant

	Description	Value
27	Iterative Imputation Categorical Model	None
28	Unknown Categoricals Handling	least_frequent
29	Normalize	True
30	Normalize Method	robust
31	Transformation	False
32	Transformation Method	None
33	PCA	False
34	PCA Method	None
35	PCA Components	None
36	Ignore Low Variance	True
37	Combine Rare Levels	False
38	Rare Level Threshold	None
39	Numeric Binning	False
40	Remove Outliers	False
41	Outliers Threshold	None
42	Remove Multicollinearity	True
43	Multicollinearity Threshold	0.9
44	Remove Perfect Collinearity	True
45	Clustering	False
46	Clustering Iteration	None
47	Polynomial Features	False
48	Polynomial Degree	None
49	Trignometry Features	False
50	Polynomial Threshold	None
51	Group Features	False
52	Feature Selection	False
53	Feature Selection Method	classic
54	Features Selection Threshold	None
55	Feature Interaction	False
56	Feature Ratio	False
57	Interaction Threshold	None
58	Fix Imbalance	False
59	Fix Imbalance Method	SMOTE

In [26]:

```
# 모델을 비교하여 최적의 모델 4개를 선택.
top4 = compare_models(n_select=4)
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lightgbm	Light Gradient Boosting Machine	0.8076	0.8998	0.8141	0.8058	0.8098	0.6151	0.6154	0.388
gbc	Gradient Boosting Classifier	0.8012	0.8949	0.8360	0.7835	0.8088	0.6022	0.6037	2.176
rf	Random Forest Classifier	0.7983	0.8816	0.7732	0.8162	0.7940	0.5967	0.5977	1.646
ada	Ada Boost Classifier	0.7963	0.8777	0.8418	0.7736	0.8061	0.5924	0.5950	0.504
lr	Logistic Regression	0.7936	0.8800	0.8293	0.7759	0.8017	0.5871	0.5885	1.712
et	Extra Trees Classifier	0.7907	0.8639	0.7610	0.8114	0.7852	0.5816	0.5829	1.372
ridge	Ridge Classifier	0.7898	0.0000	0.8506	0.7603	0.8028	0.5793	0.5836	0.066
lda	Linear Discriminant Analysis	0.7898	0.8739	0.8506	0.7603	0.8028	0.5793	0.5836	0.098
knn	K Neighbors Classifier	0.7770	0.8501	0.7658	0.7856	0.7755	0.5541	0.5543	0.662
nb	Naive Bayes	0.7727	0.8471	0.8945	0.7211	0.7984	0.5447	0.5616	0.086
dt	Decision Tree Classifier	0.7516	0.7515	0.7621	0.7486	0.7553	0.5031	0.5032	0.154
svm	SVM - Linear Kernel	0.7237	0.0000	0.7901	0.7166	0.7414	0.4468	0.4642	0.206
qda	Quadratic Discriminant Analysis	0.5085	0.5090	0.4189	0.5315	0.4490	0.0179	0.0222	0.072
dummy	Dummy Classifier	0.5031	0.5000	1.0000	0.5031	0.6694	0.0000	0.0000	0.042

In [27]:

```
print(top4[0])
```

```
LGBMClassifier(boosting_type='gbdt', class_weight=None, colsample_bytr
ee=1.0,
               importance_type='split', learning_rate=0.1, max_depth=-
1,
               min_child_samples=20, min_child_weight=0.001, min_split
_gain=0.0,
               n_estimators=100, n_jobs=-1, num_leaves=31, objective=N
one,
               random_state=7010, reg_alpha=0.0, reg_lambda=0.0, silen
t='warn',
               subsample=1.0, subsample_for_bin=200000, subsample_freq
=0)
```

In [28]:

```
# 추가 필요 라이브러리 설치
!pip install scikit-optimize
!pip install tune-sklearn ray[tune]
!pip install hpbandster ConfigSpace
!pip install optuna
```

```
Looking in indexes: https://pypi.org/simple, (https://pypi.org/simp
le,) https://us-python.pkg.dev/colab-wheels/public/simple/ (http
s://us-python.pkg.dev/colab-wheels/public/simple/)
Collecting scikit-optimize
  Downloading scikit_optimize-0.9.0-py2.py3-none-any.whl (100 kB)
     |████████████████████████████████| 100 kB 3.1 M
B/s
Requirement already satisfied: scipy>=0.19.1 in /usr/local/lib/pyth
on3.7/dist-packages (from scikit-optimize) (1.5.4)
Collecting pyaml>=16.9
  Downloading pyaml-21.10.1-py2.py3-none-any.whl (24 kB)
Requirement already satisfied: numpy>=1.13.3 in /usr/local/lib/pyth
on3.7/dist-packages (from scikit-optimize) (1.19.5)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/pytho
n3.7/dist-packages (from scikit-optimize) (1.1.0)
Requirement already satisfied: scikit-learn>=0.20.0 in /usr/local/l
ib/python3.7/dist-packages (from scikit-optimize) (0.23.2)
Requirement already satisfied: PyYAML in /usr/local/lib/python3.7/d
ist-packages (from pyaml>=16.9->scikit-optimize) (5.4.1)
```

In [29]:

```
import optuna
```

In [ ]:

```
# 참고용
# catboost_best = create_model('catboost', nan_mode= 'Min',
#                              eval_metric='Logloss',
#                              iterations=1000,
#                              sampling_frequency='PerTree',
#                              leaf_estimation_method='Newton',
#                              grow_policy='SymmetricTree',
#                              penalties_coefficient=1,
#                              boosting_type='Plain',
#                              model_shrink_mode='Constant',
#                              feature_border_type='GreedyLogSum',
#                              l2_leaf_reg=3,
#                              random_strength=1,
#                              rsm=1,
#                              boost_from_average=False,
#                              model_size_reg=0.5,
#                              subsample=0.800000011920929,
#                              use_best_model=False,
#                              class_names=[0, 1],
#                              depth=6,
#                              posterior_sampling=False,
#                              border_count=254,
#                              classes_count=0,
#                              auto_class_weights='None',
#                              sparse_features_conflict_fraction=0,
#                              leaf_estimation_backtracking='AnyImprovement',
#                              best_model_min_trees=1,
#                              model_shrink_rate=0,
#                              min_data_in_leaf=1,
#                              loss_function='Logloss',
#                              learning_rate=0.02582800015807152,
#                              score_function='Cosine',
#                              task_type='CPU',
#                              leaf_estimation_iterations=10,
#                              bootstrap_type='MVS',
#                              max_leaves=64)
```

In [30]:

```
# 성능이 좋은 특정 모델 지정.
lightgbm_best = create_model('lightgbm', nan_mode= 'Min',
                             eval_metric='Logloss',
                             iterations=1000,
                             sampling_frequency='PerTree',
                             leaf_estimation_method='Newton',
                             grow_policy='SymmetricTree',
                             penalties_coefficient=1,
                             model_shrink_mode='Constant',
                             feature_border_type='GreedyLogSum',
                             l2_leaf_reg=3,
                             random_strength=1,
                             rsm=1,
                             boost_from_average=False,
                             model_size_reg=0.5,
                             subsample=0.800000011920929,
                             use_best_model=False,
                             class_names=[0, 1],
                             depth=6,
                             posterior_sampling=False,
                             border_count=254,
                             classes_count=0,
                             auto_class_weights='None',
                             sparse_features_conflict_fraction=0,
                             leaf_estimation_backtracking='AnyImprovement',
                             best_model_min_trees=1,
                             model_shrink_rate=0,
                             min_data_in_leaf=1,
                             loss_function='Logloss',
                             learning_rate=0.02582800015807152,
                             score_function='Cosine',
                             leaf_estimation_iterations=10,
                             bootstrap_type='MVS',
                             max_leaves=64
)
```

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7938	0.8938	0.8048	0.7894	0.7970	0.5876	0.5877
1	0.8100	0.9013	0.8418	0.7933	0.8168	0.6198	0.6210
2	0.8094	0.8926	0.8303	0.7989	0.8143	0.6187	0.6192
3	0.7955	0.8912	0.8464	0.7700	0.8064	0.5907	0.5936
4	0.8181	0.9057	0.8395	0.8069	0.8229	0.6361	0.6367
Mean	0.8054	0.8969	0.8326	0.7917	0.8115	0.6106	0.6117
Std	0.0093	0.0056	0.0148	0.0124	0.0090	0.0186	0.0183

In [ ]:

```
# 다음과 같이 모델의 미세 조정이 가능.
# catboost = tune_model(create_model('catboost'), choose_better = True, n_iter = 20)
# catboost2 = tune_model(create_model('catboost'), optimize='Accuracy',
#                        search_library='scikit-optimize', search_algorithm='bayesia
#                        choose_better = True, n_iter = 20)
# catboost3 = tune_model(create_model('catboost'), optimize='Accuracy',
#                        search_library='tune-sklearn', search_algorithm='bayesian',
#                        choose_better = True, n_iter = 20)
# catboost4 = tune_model(create_model('catboost'), optimize='Accuracy',
#                        search_library='tune-sklearn', search_algorithm='hyperopt',
#                        choose_better = True, n_iter = 20)
# catboost5 = tune_model(create_model('catboost'), optimize='Accuracy',
#                        search_library='tune-sklearn', search_algorithm='optuna',
#                        choose_better = True, n_iter = 20)
# catboost6 = tune_model(create_model('catboost'), optimize='Accuracy',
#                        search_library='optuna', search_algorithm='tpe',
#                        choose_better = True, n_iter = 20)
```

앙상블(Ensemble)

In [ ]:

```
# tuned_top4 = [tune_model(i) for i in top4]
# blender_top4 = blend_models(estimator_list=tuned_top4)
```

In [31]:

```
# 지정된 모델을 이용하여 예측을 수행
df_pred = predict_model(lightgbm_best, X_test)
y_pred = df_pred.loc[:, ['Label']]
```

모델 해석

In [32]:

```
submission['Transported'] = y_pred
submission.to_csv('submission.csv', index=False)
submission
```

Out[32]:

	PassengerId	Transported
0	0013_01	True
1	0018_01	False
2	0019_01	True
3	0021_01	True
4	0023_01	False
...	...	...
4272	9266_02	True
4273	9269_01	False
4274	9271_01	True
4275	9273_01	True
4276	9277_01	True

4277 rows × 2 columns

제출 결과 : 0.80454

REF

- gc 에 대해 이해해 보기
  - https://medium.com/dmsfordsm/garbage-collection-in-python-777916fd3189 (https://medium.com/dmsfordsm/garbage-collection-in-python-777916fd3189)