2. 전처리 미션(1): 호텔 객실 취소 예측

주어진 예약 데이터를 통해 취소될 객실인지 예측을 하는 분석 모델을 생성해봅니다.

👀 **데이터 컬럼 소개**

컬럼명	설명
is_canceled	예약이 취소되었는지 여부를 나타내는 이진 변수 (0: 비취소, 1: 취소)
lead_time	예약 날짜와 실제 도착 날짜 사이의 일수
stays_in_weekend_nights	주말 동안 머무는 밤의 수
stays_in_week_nights	평일 동안 머무는 밤의 수
is_repeated_guest	반복 방문자인지 여부를 나타내는 이진 변수 (0: 처음 방문, 1: 반복 방문)
previous_cancellations	과거에 취소한 예약 횟수
previous_bookings_not_canceled	과거에 취소하지 않은 예약 횟수
booking_changes	예약 후 변경된 횟수
days_in_waiting_list	대기 명단에 있는 일수
adr	평균 일일 요금 (Average Daily Rate)
deposit_type	보증금의 종류를 나타내는 범주형 변수

2.1 EDA(데이터 탐색)

2.1.1 데이터 불러오기

In [ ]:
```
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# 데이터 불러오기
hotel = pd.read_csv("./data/hotel.csv")
# 데이터 확인
hotel.head()
```

Out[ ]:

	is_canceled	deposit_type	lead_time	stays_in_weekend_nights	stays_in_week_nights	is
0	0	No Deposit	105.0	2	5	
1	0	No Deposit	303.0	2	2	
2	0	No Deposit	33.0	2	3	
3	0	No Deposit	48.0	0	1	
4	0	No Deposit	216.0	4	7	

2.1.2 데이터 개요 확인하기

💥 MISSION

> Dataframe의 info, describe 기능을 활용하여 주어진 데이터를 해석해보기

In [ ]:
```
hotel_info = hotel.info()
hotel_info
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20000 entries, 0 to 19999
Data columns (total 11 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   is_canceled                     20000 non-null  int64
 1   deposit_type                    20000 non-null  object
 2   lead_time                       19995 non-null  float64
 3   stays_in_weekend_nights         20000 non-null  int64
 4   stays_in_week_nights            20000 non-null  int64
 5   is_repeated_guest               19642 non-null  float64
 6   previous_cancellations          20000 non-null  int64
 7   previous_bookings_not_canceled  20000 non-null  int64
 8   booking_changes                 20000 non-null  int64
 9   days_in_waiting_list            20000 non-null  int64
 10  adr                             18937 non-null  float64
dtypes: float64(3), int64(7), object(1)
memory usage: 1.7+ MB
```

In [ ]:
```
hotel_desc = hotel.describe()
hotel_desc
```

Out[ ]:

	is_canceled	lead_time	stays_in_weekend_nights	stays_in_week_nights	is_repe
count	20000.00000	19995.000000	20000.000000	20000.000000	19
mean	0.12000	85.978345	0.892550	2.380400	
std	0.32497	96.427240	0.952077	1.777345	
min	0.00000	0.000000	0.000000	0.000000	
25%	0.00000	11.000000	0.000000	1.000000	
50%	0.00000	51.000000	1.000000	2.000000	
75%	0.00000	132.000000	2.000000	3.000000	
max	1.00000	629.000000	13.000000	30.000000	

🔬 데이터 요약 결과 해석

1. 📌 **is_canceled**
   - (값) 0-예약이 취소되지 않음/ 1-예약이 취소 됨
   - (평균값) 0.12
   - ☞ **전체 예약 중 약 12%가 취소됨을 의미하고 이는 target 데이터의 불균형을 뜻함**
2. **lead_time**
   - (결측치) 5개
   - (평균) 85.98일/ (최소값)0일/ (최대값) 629일
   - ☞ **의미상 2년전 예약하는 경우는 없을 것 같기에 이상치 판단 필요**
3. **stays_in_weekend_nights**

- (평균) 0.89일/ (최대값) 13일

4. **stays_in_week_nights**

- (평균) 2.38일/ (최대값) 30일

5. **is_repeated_guest**

- (값) 0-첫 고객/ 1-기존 고객
- (결측치) 358개
- (평균) 3.8
- ☞ **예약 고객 중 약 3.8%가 기존 고객임을 의미하고 이는 feature 데이터의 불균형을 뜻함**

6. **previous_cancellations**

- (평균) 0.03회/ (최대값) 26회
- ☞ **취소를 26번한 것은 의미적으로도 어색하고 평균과 차이가 많이 나기 때문에 이상치인지 판단 필요**

7. **previous_bookings_not_canceled**

- (평균) 0.17회/ (최대값) 66회

8. **booking_changes**

- (평균) 0.27회/ (최대값) 17회

9. **days_in_waiting_list**

- (평균) 1.98일/ (최대값) 379일
- ☞ **평균과 차이가 많이 나기 때문에 이상치인지 판단 필요**

10. **adr (Average Daily Rate)**

- (결측치) 1063개
- (평균) 101.41/ (최소값) -6.38/ (최대값) 451.50
- ☞ **요금이 음수일 수 없으니 이상치 확실함**

11. **deposit_type**

- ☞ **범주형 데이터이기에 고유값 확인 필요**

✅ 예측 목적인 is_cancled가 범주형이기에 분류(classificiation) 분석을 수행함

2.1.3 시각화

1. 수치형 변수 시각화

💥 **MISSION**

> list에 수치형 변수들을 정의해보기

In [ ]:
```
# 수치형
continuous_columns = ['lead_time', 'stays_in_weekend_nights', 'stays_in_week
                      'previous_cancellations', 'previous_bookings_not_cance
                      'booking_changes', 'days_in_waiting_list', 'adr']
```

📍 **히스토그램**

- 수치형 변수들을 히스토그램으로 시각하여 이상치에 대한 판단

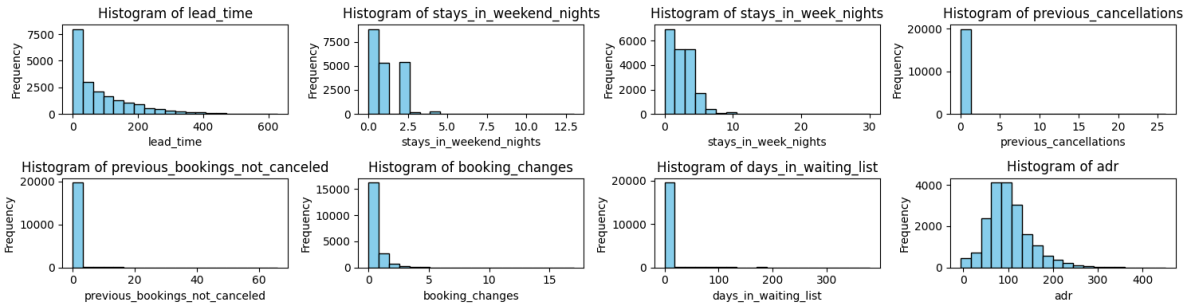
In [ ]:
```
fig, axes = plt.subplots(nrows=2, ncols=4, figsize=(15, 4))

for i, column in enumerate(continuous_columns):

    row = i // 4
    col = i % 4
    hotel[column].plot(kind='hist', bins=20, ax=axes[row, col], color='skybl
    axes[row, col].set_title(f'Histogram of {column}')
    axes[row, col].set_xlabel(column)
    axes[row, col].set_ylabel('Frequency')

plt.tight_layout()

plt.show()
```



📍 상자 그림

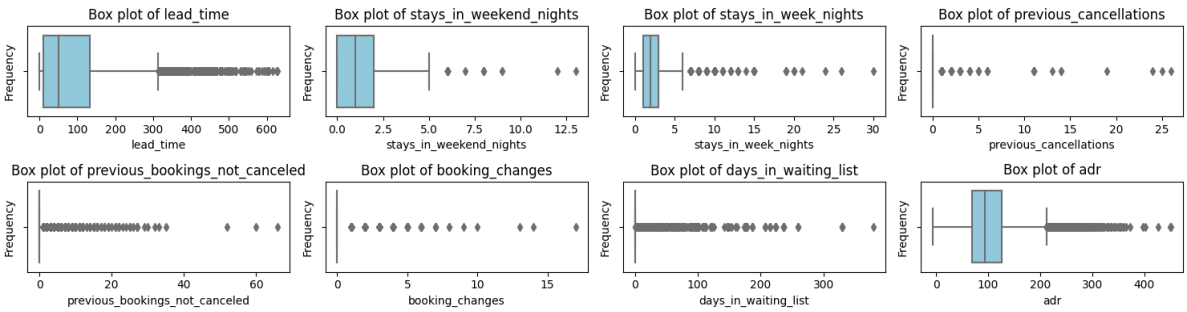
- 히스토그램으로 분포를 파악해보았지만 이상치를 완벽하게 구분해 낼 수 없기에 상자 그림으로 이상치를 명확히 판단

In [ ]:
```
fig, axes = plt.subplots(nrows=2, ncols=4, figsize=(15, 4))

for i, column in enumerate(continuous_columns):

    row = i // 4
    col = i % 4
    sns.boxplot(x=hotel[column], ax=axes[row, col], color='skyblue')
    axes[row, col].set_title(f'Box plot of {column}')
    axes[row, col].set_xlabel(column)
    axes[row, col].set_ylabel('Frequency')

plt.tight_layout()
plt.show()
```



🔬 수치형 변수 시각화 결과 해석

- 상자 그림으로 표현시 모든 변수에서 이상치를 포함하고 있음

- 데이터 상으로는 이상치로 보이나 연속적으로 나타나있기 때문에 업무적으로는 이상치가 아닐 수 있음
- 각 변수에서 너무 동떨어진 몇개의 포인트만을 제거하는 보수적인 이상치 제거 방식 적용

2. 범주형 변수 시각화

💥 MISSION

> list에 범주형 변수들을 정의해보기

In [ ]:
```
categorical_columns = ['is_canceled', 'deposit_type', 'is_repeated_guest']
```

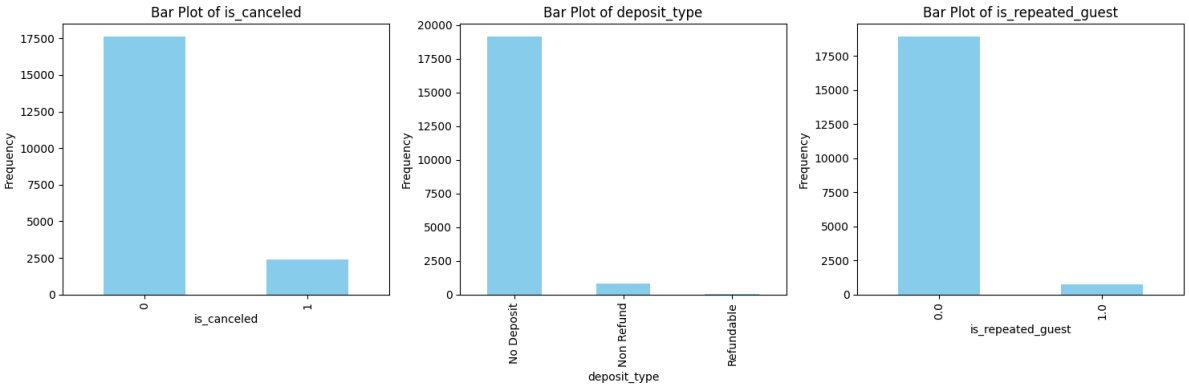
📍 막대그래프

In [ ]:
```
fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(15, 5))

for i, column in enumerate(categorical_columns):

    hotel[column].value_counts().sort_index().plot(kind='bar', ax=axes[i], c
    axes[i].set_title(f'Bar Plot of {column}')
    axes[i].set_xlabel(column)
    axes[i].set_ylabel('Frequency')

plt.tight_layout()
plt.show()
```



🔬 범주형 변수 시각화 결과 해석

- 모든 범주형 변수가 불균형임을 확인
- deposit_type은 범주형 변수처리 필요

3. 그룹별 평균 시각화

- 분류 문제일 경우 그룹별로 데이터를 시각화하여 파악하는 작업이 효과적일 수 있음

In [ ]:
```
# 그룹별 평균
is_canceled = hotel.groupby('is_canceled').mean()
is_canceled
```

Out[ ]:

is_canceled	lead_time	stays_in_weekend_nights	stays_in_week_nights	is_repeated_guest
0	78.420290	0.888750	2.361080	0.041481
1	141.388333	0.920417	2.522083	0.013577

📍 그룹별 막대 그래프

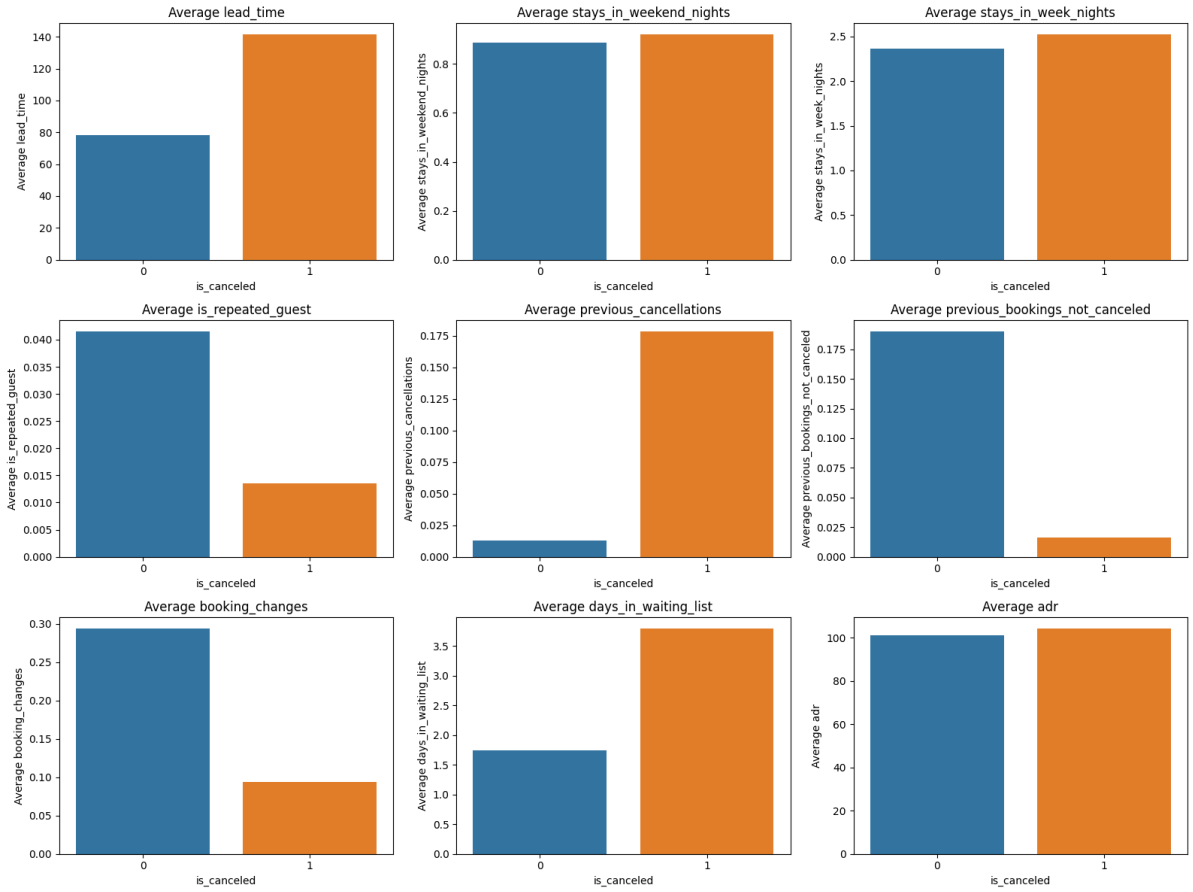
In [ ]:

```
# 그룹별 막대 그래프
fig, axes = plt.subplots(3, 3, figsize=(16, 12))

columns = is_canceled.columns.to_list()

for i, column in enumerate(columns):
    row = i // 3
    col = i % 3
    sns.barplot(x=is_canceled.index, y=column, data=is_canceled, ax=axes[row
    axes[row, col].set_title(f'Average {column}')
    axes[row, col].set_xlabel("is_canceled")
    axes[row, col].set_ylabel(f'Average {column}')

plt.tight_layout()
plt.show()
```



🔬 그룹별 평균 시각화 결과 해석

- 예약 취소하지 않음/ 예약 취소 간의 그룹별 평균이 차이가 클 수록 분류에 영향을 많이 미칠 것으로 보임
- 모델의 성능이 좋지 못하면 그룹간 평균 차이가 없는 데이터를 피쳐에서 삭제해보는 방법 고려

2.2 데이터 전처리

2.2.1 결측치 탐색 및 처리

1. 결측치 탐색

💥 MISSION

> 컬럼별 결측치의 개수를 확인해보자

In [ ]:
```
hotel.isna().sum()
```

Out[ ]:
```
is_canceled                         0
deposit_type                        0
lead_time                           5
stays_in_weekend_nights             0
stays_in_week_nights                0
is_repeated_guest                 358
previous_cancellations              0
previous_bookings_not_canceled      0
booking_changes                     0
days_in_waiting_list                0
adr                              1063
dtype: int64
```

📍 결측치 비율 확인

In [ ]:
```
for col in hotel.columns:
    if hotel[col].isna().sum() > 0:
        print("*"*15)
        print("결측치 컬럼 :", col)
        print("결측치 비율 : {}%".format(hotel[col].isna().sum()/len(hotel)*100
```

```
***************
결측치 컬럼 : lead_time
결측치 비율 : 0.025%
***************
결측치 컬럼 : is_repeated_guest
결측치 비율 : 1.79%
***************
결측치 컬럼 : adr
결측치 비율 : 5.315%
```

🔬 결측치 탐색 결과 해석

- lead_time 결측치 비율이 작기에 제거해도 무방해 보임
- is_repeated_guest는 시각화를 통해 대부분이 0이 었으니 0으로 대체하여도 무방해 보임
- adr은 수치형이기 때문에 평균값으로 대체

2. 결측치 처리

💥 MISSION

> lead_time, is_repeated_guest, adr 컬럼에 대해 근거를 가지고 결측치 처리해보기

In [ ]:
```
hotel.dropna(subset=['lead_time'], axis=0, inplace=True)
hotel['is_repeated_guest'].fillna(0, inplace = True)
```

```
hotel['adr'].fillna(hotel['adr'].mean(), inplace = True)
```

- lead_time은 결측치의 비율이 낮기 때문에 단순 삭제
- is_repeated_guset는 시각화 결과 대부분이 0이었기에 대체
- adr은 수치형이기 때문에 평균으로 대체

📍 **결측치 처리 결과**

In [ ]:
```
hotel.isna().sum()
```

Out[ ]:
```
is_canceled                     0
deposit_type                    0
lead_time                       0
stays_in_weekend_nights         0
stays_in_week_nights            0
is_repeated_guest               0
previous_cancellations          0
previous_bookings_not_canceled  0
booking_changes                 0
days_in_waiting_list            0
adr                             0
dtype: int64
```

2.2.2 이상치 탐색 및 처리

1. 이상치 탐색

📍 **IQR**

In [ ]:
```
iqr_outliers = pd.DataFrame()

for column in continuous_columns:
    Q1 = hotel[column].quantile(0.25)
    Q3 = hotel[column].quantile(0.75)
    IQR = Q3 - Q1
    # IQR 기준으로 이상치 데이터 필터링
    outliers_in_column = hotel[((hotel[column] < (Q1 - 1.5 * IQR)) | (hotel[
    # 이상치 데이터를 iqr_outliers 데이터프레임에 추가
    iqr_outliers = pd.concat([iqr_outliers, outliers_in_column], axis=0).drc

iqr_outliers
```

Out[ ]:

	is_canceled	deposit_type	lead_time	stays_in_weekend_nights	stays_in_week_nigh
14	0	No Deposit	377.0	0	
85	0	No Deposit	422.0	0	
164	0	No Deposit	316.0	0	
170	0	No Deposit	377.0	0	
227	0	No Deposit	338.0	1	
...	...	...	...	...	
19848	1	No Deposit	91.0	2	
19852	1	No Deposit	102.0	2	
19938	1	No Deposit	19.0	0	
19944	1	No Deposit	50.0	0	
19998	1	No Deposit	0.0	0	

5365 rows × 11 columns

📍 상위 n%

In [ ]:

```
outliers = pd.DataFrame()

for column in continuous_columns:
    # 각 연속형 변수에 대해 상위 95%에 해당하는 임계값 계산
    threshold = hotel[column].quantile(0.99)
    # 임계값보다 큰 데이터만 필터링하여 이상치로 판단
    outliers_in_column = hotel[hotel[column] > threshold]
    # 이상치 데이터를 outliers 데이터프레임에 추가
    outliers = pd.concat([outliers, outliers_in_column], axis=0).drop_duplic
outliers
```

Out[ ]:

	is_canceled	deposit_type	lead_time	stays_in_weekend_nights	stays_in_week_nigh
85	0	No Deposit	422.0	0	
279	0	No Deposit	518.0	2	
355	0	No Deposit	434.0	2	
399	0	No Deposit	414.0	0	
670	0	No Deposit	451.0	0	
...	...	...	...	...	
19686	1	No Deposit	22.0	2	
19800	1	No Deposit	295.0	1	
19811	1	No Deposit	20.0	0	
19848	1	No Deposit	91.0	2	
19852	1	No Deposit	102.0	2	

880 rows × 11 columns

💥 MISSION

> IQR 방식과 n% 방식의 이상치 비율을 확인해보기

In [ ]:
```
print("이상치 비율 iqr:", len(iqr_outliers)/len(hotel)*100)
print("이상치 비율 n%:", len(outliers)/len(hotel)*100)
```

```
이상치 비율 iqr: 26.831707926981746
이상치 비율 n%: 4.401100275068767
```

🔬 **이상치 탐색 결과 해석**

- 시각화를 통해 이상치로 보여도 데이터들이 연속적으로 존재하여 업무적으로는 이상치가 아닐 수 있음을 확인
- iqr 방식을 수행하면 너무 많은 데이터가 제거됨
- 비즈니스 로직을 모르기 때문에 보수적으로 이상치 제거

2. 이상치 제거

In [ ]:
```
print("원본 데이터셋 개수", len(hotel))
for column in continuous_columns:
    # 각 연속형 변수에 대해 상위 percent 95%에 해당하는 임계값 계산
    threshold = hotel[column].quantile(0.99)
    # 임계값보다 작은 데이터만 필터링
    hotel = hotel[hotel[column] <= threshold]
print("이상치 처리 후 개수", len(hotel))
```

```
원본 데이터셋 개수 19995
이상치 처리 후 개수 18885
```

2.2.3 범주형 변수 처리

1. 인코딩

💥 **MISSION**

> deposit_type에 대해 알맞은 범주형 변수 처리 해보기

In [ ]:
```
# hotel은 순서형 변수가 아닌 명목형 변수이므로 원핫인코딩을 사용하여 변환

from sklearn.preprocessing import OneHotEncoder

# 원핫 인코딩 객체 생성 및 학습
oh = OneHotEncoder(sparse_output=False)
oh.fit(hotel[['deposit_type']])

# 원핫 인코딩 수행
oh_encoded = oh.transform(hotel[['deposit_type']])

# 데이터프레임으로 변환, 열 이름 지정
oh_hotel = pd.DataFrame(oh_encoded.astype('int'), columns=oh.get_feature_nam

oh_hotel
```

Out[ ]:

	deposit_type_No Deposit	deposit_type_Non Refund	deposit_type_Refundable
0	1	0	0
1	1	0	0
2	1	0	0
3	1	0	0
4	1	0	0
...	...	...	...
19995	0	1	0
19996	0	1	0
19997	0	1	0
19998	1	0	0
19999	0	1	0

18885 rows × 3 columns

2. 데이터 병합

💥 MISSION

> hotel 데이터에서 변환을 수행한 deposit_type 컬럼은 삭제하고, 인코딩 수행한 변수를 열 기준으로 concat 해보기

In [ ]:
```
hotel = pd.concat([hotel.drop('deposit_type', axis=1), oh_hotel], axis=1)
hotel
```

Out[ ]:

	is_canceled	lead_time	stays_in_weekend_nights	stays_in_week_nights	is_repeated
0	0	105.0	2	5	
1	0	303.0	2	2	
2	0	33.0	2	3	
3	0	48.0	0	1	
4	0	216.0	4	7	
...	...	...	...	...	
19995	1	89.0	2	2	
19996	1	101.0	0	3	
19997	1	277.0	1	2	
19998	1	0.0	0	1	
19999	1	40.0	0	2	

18885 rows × 13 columns

🔬 범주형 변수처리 결과 해석

- deposit_type에 3개의 범주형 데이터가 있었고, 이를 원핫 인코딩을 통해 변환

- 3개의 범주형 데이터를 3개의 컬럼으로 나타내게 변환하고 이를 원본 데이터에 병합

2.2.4 데이터 분할

💥 MISSION

> hotel 데이터를 train(0.7)/ validation(0.2)/ test(0.1) 비율로 분할 해보기

In [ ]:
```
from sklearn.model_selection import train_test_split

# 데이터 분할
train_data, test_data = train_test_split(hotel, test_size = 0.2, random_stat
validation_data, test_data = train_test_split(test_data, test_size = 0.5, ra

train_data.reset_index(drop=True, inplace=True)
validation_data.reset_index(drop=True, inplace=True)
test_data.reset_index(drop=True, inplace=True)

print('train data :', train_data.shape)
print('validation data :', validation_data.shape)
print('test data :', test_data.shape)
```

```
train data : (15108, 13)
validation data : (1888, 13)
test data : (1889, 13)
```

2.2.5 데이터 스케일링

1. 데이터 범위 확인

💥 MISSION

> train_data의 요약 통계 정보를 확인해보기

In [ ]:
```
train_data.describe()
```

Out[ ]:

	is_canceled	lead_time	stays_in_weekend_nights	stays_in_week_nights	is_repe
count	15108.000000	15108.000000	15108.000000	15108.000000	15
mean	0.116428	80.966773	0.865568	2.289515	
std	0.320749	88.131655	0.880643	1.518703	
min	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	10.000000	0.000000	1.000000	
50%	0.000000	49.000000	1.000000	2.000000	
75%	0.000000	125.000000	2.000000	3.000000	
max	1.000000	409.000000	4.000000	8.000000	

🔬 데이터 범위 확인 결과 해석

- lead_time, days_in_waiting_list, adr 변수가 다른 값에 비해 큰값을 가지고 있음을 확인

2. 스케일링 수행(훈련 데이터)

MISSION

> train_data에 알맞은 스케일링 수행해보기

In [ ]:
```
# 분류 분석이기에 standard scaling을 사용하여 데이터 스케일링
from sklearn.preprocessing import StandardScaler

scale_columns = ['lead_time', 'days_in_waiting_list', 'adr']

StdScaler = StandardScaler()
# Train 데이터의 fitting과 스케일링
StdScaler.fit(train_data[scale_columns])
scaled_data = StdScaler.transform(train_data[scale_columns])

# 데이터프레임 변환
scaled_df = pd.DataFrame(scaled_data, columns=scale_columns)
scaled_train_data = pd.concat([train_data.drop(scale_columns, axis=1), scale
scaled_train_data.head()
```

Out[ ]:

	is_canceled	stays_in_weekend_nights	stays_in_week_nights	is_repeated_guest	previou
0	1	0	3	0.0	
1	0	1	3	0.0	
2	0	2	5	0.0	
3	0	1	5	0.0	
4	1	2	2	0.0	

3. 스케일링 적용(검증, 테스트 데이터)

In [ ]:
```
# 검증용 데이터 스케일링 적용
scaled_data = StdScaler.transform(validation_data[scale_columns])
scaled_df = pd.DataFrame(scaled_data, columns=scale_columns)
scaled_val_df = pd.concat([validation_data.drop(scale_columns, axis=1), scal

scaled_val_df.head()
```

Out[ ]:

	is_canceled	stays_in_weekend_nights	stays_in_week_nights	is_repeated_guest	previou
0	0	2	1	0.0	
1	0	2	3	0.0	
2	1	2	3	0.0	
3	1	0	3	0.0	
4	0	0	1	0.0	

In [ ]:
```
# 테스트 데이터 스케일링 적용
scaled_data = StdScaler.transform(test_data[scale_columns])
scaled_df = pd.DataFrame(scaled_data, columns=scale_columns)
scaled_test_df = pd.concat([test_data.drop(scale_columns, axis=1), scaled_df

scaled_test_df.head()
```

Out[ ]:

	is_canceled	stays_in_weekend_nights	stays_in_week_nights	is_repeated_guest	previou
0	0	0	3	0.0	
1	0	1	1	0.0	
2	0	2	5	0.0	
3	0	0	2	1.0	
4	0	2	2	0.0	

2.2.6 데이터 불균형 문제 처리

1. Target 불균형 확인

💥 **MISSION**

> train_data에서 Target인 is_canceled의 비율을 확인해보기

In [ ]:
```
ratio0 = round(len(train_data[train_data['is_canceled']==0])/len(train_data)
ratio1 = round(len(train_data[train_data['is_canceled']==1])/len(train_data)
print('0 비율: {}%'.format(ratio0))
print('1 비율: {}%'.format(ratio1))
```

```
0 비율: 88.36%
1 비율: 11.64%
```

2. 오버샘플링 수행

💥 **MISSION**

> is_canceled 컬럼을 기준으로 오버샘플링 수행해보기

In [ ]:
```
from imblearn.over_sampling import RandomOverSampler
from collections import Counter

# 오버샘플링 수행
oversample = RandomOverSampler()

# 데이터 분할
x = scaled_train_data.drop('is_canceled', axis=1)
y = scaled_train_data['is_canceled']

x_over, y_over = oversample.fit_resample(x, y)
print(Counter(y_over))
```

```
Counter({1: 13349, 0: 13349})
```

🔬 **오버샘플링 결과 해석**

- 오버샘플링을 통해 0과 1의 비율을 동등하게 만듦

2.3 예측 모델 생성

2.3.1 선형 모델 생성 및 학습 (통계기반)

📍 **Support Vector Classification**

In [ ]:
```
from sklearn.svm import SVC

svc = SVC() # 선형 커널 사용
svc.fit(x_over, y_over)

print('train 정확도 :', svc.score(x_over, y_over))
```

```
Intel MKL WARNING: Support of Intel(R) Streaming SIMD Extensions 4.2 (Intel
(R) SSE4.2) enabled only processors has been deprecated. Intel oneAPI Math
Kernel Library 2025.0 will require Intel(R) Advanced Vector Extensions (Int
el(R) AVX) instructions.
train 정확도 : 0.712937298674058
```

2.3.2 트리기반 모델 생성 및 학습

📍 **RandomForestClassifier**

In [ ]:
```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(random_state=1004)
rf.fit(x_over, y_over)

print('train 정확도 :', rf.score(x_over, y_over))
```

```
train 정확도 : 0.99411940969361
```

2.4 분석 모델 성능 평가

분석 수행시 validation 데이터로 평가를 하며 모델의 파라미터 혹은 변수를 수정하는 튜닝 과정을 거친 뒤, test 데이터로 평가합니다. 이번 실습에서는 튜닝은 생략하고 validation 데이터와 test 데이터를 함께 test 데이터로 구성하여 최종 평가를 수행합니다.

1. 데이터 병합

In [ ]:
```
# 행 기준으로 병합
total_test_data = pd.concat([scaled_val_df, scaled_test_df], axis=0)
total_test_data.reset_index(drop=True, inplace=True)
total_test_data.head()
```

Out[ ]:

	is_canceled	stays_in_weekend_nights	stays_in_week_nights	is_repeated_guest	previou
0	0	2	1	0.0	
1	0	2	3	0.0	
2	1	2	3	0.0	
3	1	0	3	0.0	
4	0	0	1	0.0	

2. 데이터 분할

In [ ]:
```
X_test = total_test_data.drop('is_canceled', axis=1)
y_test = total_test_data['is_canceled'].values
```

2.4.1 성능 평가(1)

📍 Confusion Matrix

In [ ]:
```
from sklearn.metrics import classification_report

# svc 모델 성능 평가
print('SVC 성능평가')
y_pred=svc.predict(X_test)
print(classification_report(y_test, y_pred))
print('test 정확도 :', svc.score(X_test, y_test))

# rf 모델 성능 평가
print('RF 성능평가')
y_pred=rf.predict(X_test)
print(classification_report(y_test, y_pred))
print('test 정확도 :', rf.score(X_test, y_test))
```

```
SVC 성능평가
              precision    recall  f1-score   support

           0       0.94      0.79      0.86      3337
           1       0.28      0.62      0.38       440

    accuracy                           0.77      3777
   macro avg       0.61      0.70      0.62      3777
weighted avg       0.86      0.77      0.80      3777

test 정확도 : 0.7688641779189833
RF 성능평가
              precision    recall  f1-score   support

           0       0.93      0.96      0.94      3337
           1       0.57      0.44      0.50       440

    accuracy                           0.90      3777
   macro avg       0.75      0.70      0.72      3777
weighted avg       0.89      0.90      0.89      3777

test 정확도 : 0.8967434471803019
```

🔬 Confusion Matrix 성능평가 결과 해석

1. Accuracy (정확도): 0.89

- 전체 3777개의 샘플 중 89%를 모델이 올바르게 예측했습니다.

2. Macro Average: 각 클래스의 성능을 동일하게 고려한 평균입니다. 클래스 1의 성능이 낮아, 전체적인 성능이 낮아집니다.

- Precision: 0.74
- Recall: 0.70
- F1-score: 0.72

3. Weighted Average: 각 클래스의 샘플 수를 가중치로 사용하여 계산된 평균입니다. 클래스 0의 샘플 수가 많기 때문에, 클래스 0의 성능이 전반적인 평균에 더 큰 영향을 미칩니다.

- Precision: 0.88

- Recall: 0.89
- F1-score: 0.89

2.4.2 성능 평가(2)

📍 ROC, AUC

In [ ]:
```
from sklearn.metrics import roc_curve, auc

# 테스트 데이터에 대한 예측 확률 계산
y_pred_prob = rf.predict_proba(X_test)[:, 1]

# ROC 커브 계산
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)

# AUC 계산
roc_auc = auc(fpr, tpr)

# ROC 커브 그리기
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()
```