호텔 예약 취소 여부 예측_project

진행 일정

2022.11.22~2022.11.29

Project 설명

소비자가 호텔 예약을 취소할 지 실제 이용할 지를 예측 > 이진 분류

0. 데이터 설명

필요한 라이브러리 설치

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

from sklearn.inspection import permutation_importance

!pip install category_encoders
import category_encoders as ce

from sklearn.model_selection import train_test_split

from collections import Counter

from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
from sklearn.ensemble import AdaBoostClassifier

from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

from sklearn.ensemble import VotingClassifier

import warnings
warnings.filterwarnings('ignore')
```

변수 설명

hotel	호텔 종류 / 문자형	credit_card	신용카드 고유번호
lead_time	몇 일 전에 예약했는지 / 숫자	reserved_room_type	예약한 룸 타입
arrival_date_year	도착 연도 / 숫자	assigned_room_type	실제 배정받은 룸 타입
arrival_date_month	도착 월 / 문자	booking_changes	예약 변경 여부
arrival_date_week_number	도착 주 차	deposit_types	예치금
arrival_date_day_of_month	도착 일	agent	예약대행 회사
stays_in_weekend_nights	주말이 몇 일 포함되어있는지	phone_number	핸드폰 번호

stays_in_week_nights	평일이 몇 일 포함되어있는지	Email	고유 이메일
adults	성인의 수	Name	이름
children	미성년자의 수	Reservation_status_date	reservation_status를 언제 고객이 행동하였는지
babies	아이의 수	previous_bookings_not_canceld	과거 취소하지 않은 이력
meal	BB(Bed&Breakfast) / HB(Half Board) / FB(Full Board) / SC(파악 불가ㅠ) / Undefined	total_of_special_requests	고객이 특수한 요청을 한 횟수(ex twin bed, high floor)
country	국적	Required_car_parking_spaces	고객이 요청한 주차 공간 갯수
market_segment	어떤 유통 채널으로 예약했는지(TA" : "Travel Agents" , "TO" : "Tour Operators")	Adr	1일 평균 요금(모든 숙박 거래의 합계를 총 숙박일수로 나누어 계산)
distribution_channel	어떤 유통 채널으로 예약했는지(TA" : "Travel Agents" , "TO" : "Tour Operators")	Customer_type	transient / transient-party / contract / group
is_repeated_guest	과거 방문 여부	Days_in_waiting_list	예약이 고객에게 확인되기 전까지 대기자 명단에 있었던 일 수
previous_cancellations	과거 취소 이력	Company	예약을 하였거나 예약을 지불할 책임이 있는 '회사/단체'의 ID

데이터의 Feature는 34개이고, target변수는 "is_canceled"

데이터 확인

```
train = pd.read_csv('kuggle_train.csv')
train.info()
train = train.drop(columns=['Unnamed: 0'],axis=1)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 76409 entries, 0 to 76408
Data columns (total 35 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Unnamed: 0                      76409 non-null  int64
 1   hotel                           76409 non-null  object
 2   lead_time                       76409 non-null  int64
 3   arrival_date_year               76409 non-null  int64
 4   arrival_date_month              76409 non-null  object
 5   arrival_date_week_number        76409 non-null  int64
 6   arrival_date_day_of_month       76409 non-null  int64
 7   stays_in_weekend_nights         76409 non-null  int64
 8   stays_in_week_nights            76409 non-null  int64
 9   adults                          76409 non-null  int64
 10  children                        76406 non-null  float64
 11  babies                          76409 non-null  int64
 12  meal                            76409 non-null  object
 13  country                         76098 non-null  object
 14  market_segment                  76409 non-null  object
 15  distribution_channel            76409 non-null  object
 16  is_repeated_guest               76409 non-null  int64
 17  previous_cancellations          76409 non-null  int64
 18  previous_bookings_not_canceled  76409 non-null  int64
 19  reserved_room_type              76409 non-null  object
 20  assigned_room_type              76409 non-null  object
 21  booking_changes                 76409 non-null  int64
 22  deposit_type                    76409 non-null  object
 23  agent                           65885 non-null  float64
 24  company                         4312 non-null   float64
 25  days_in_waiting_list            76409 non-null  int64
 26  customer_type                   76409 non-null  object
 27  adr                             76409 non-null  float64
 28  required_car_parking_spaces     76409 non-null  int64
 29  total_of_special_requests       76409 non-null  int64
 30  name                            76409 non-null  object
 31  email                           76409 non-null  object
 32  phone-number                    76409 non-null  object
 33  credit_card                     76409 non-null  object
 34  is_canceled                     76409 non-null  int64
dtypes: float64(4), int64(17), object(14)
memory usage: 20.4+ MB
```

```
test = pd.read_csv('kuggle_test.csv')
test.info()
test = test.drop(columns=['Unnamed: 0'],axis=1)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19102 entries, 0 to 19101
Data columns (total 34 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Unnamed: 0                      19102 non-null  int64
 1   hotel                           19102 non-null  object
 2   lead_time                       19102 non-null  int64
 3   arrival_date_year               19102 non-null  int64
 4   arrival_date_month              19102 non-null  object
 5   arrival_date_week_number        19102 non-null  int64
 6   arrival_date_day_of_month       19102 non-null  int64
 7   stays_in_weekend_nights         19102 non-null  int64
 8   stays_in_week_nights            19102 non-null  int64
 9   adults                          19102 non-null  int64
 10  children                        19101 non-null  float64
 11  babies                          19102 non-null  int64
 12  meal                            19102 non-null  object
 13  country                         19027 non-null  object
 14  market_segment                  19102 non-null  object
 15  distribution_channel            19102 non-null  object
 16  is_repeated_guest               19102 non-null  int64
 17  previous_cancellations          19102 non-null  int64
 18  previous_bookings_not_canceled  19102 non-null  int64
 19  reserved_room_type              19102 non-null  object
 20  assigned_room_type              19102 non-null  object
 21  booking_changes                 19102 non-null  int64
 22  deposit_type                    19102 non-null  object
 23  agent                           16509 non-null  float64
 24  company                         1117 non-null   float64
 25  days_in_waiting_list            19102 non-null  int64
 26  customer_type                   19102 non-null  object
 27  adr                             19102 non-null  float64
 28  required_car_parking_spaces     19102 non-null  int64
 29  total_of_special_requests       19102 non-null  int64
 30  name                            19102 non-null  object
 31  email                           19102 non-null  object
 32  phone-number                    19102 non-null  object
 33  credit_card                     19102 non-null  object
dtypes: float64(4), int64(16), object(14)
memory usage: 5.0+ MB
```

1. 전처리

결측치 확인

```
# train 결측치 확인 -> 결측치 확인하는 함수 생성
def check_missing_col(dataframe):
    missing_col = []
    counted_missing_col = 0
    for i, col in enumerate(dataframe.columns):
        missing_values = sum(dataframe[col].isna())
        is_missing = True if missing_values >= 1 else False
        if is_missing:
            counted_missing_col += 1
            print(f'결측치가 있는 컬럼은: {col}입니다')
            print(f'해당 컬럼에 총 {missing_values}개의 결측치가 존재합니다.')
            missing_col.append([col, dataframe[col].dtype])
    if counted_missing_col == 0:
        print('결측치가 존재하지 않습니다')
    return missing_col

missing_col = check_missing_col(train)
```

```
결측치가 있는 컬럼은: children입니다
해당 컬럼에 총 3개의 결측치가 존재합니다.
결측치가 있는 컬럼은: country입니다
해당 컬럼에 총 311개의 결측치가 존재합니다.
결측치가 있는 컬럼은: agent입니다
해당 컬럼에 총 10524개의 결측치가 존재합니다.
결측치가 있는 컬럼은: company입니다
해당 컬럼에 총 72097개의 결측치가 존재합니다.
```

```
# test 결측치 확인
missing_col = check_missing_col(test)
```

```
결측치가 있는 컬럼은: children입니다
해당 컬럼에 총 1개의 결측치가 존재합니다.
결측치가 있는 컬럼은: country입니다
해당 컬럼에 총 75개의 결측치가 존재합니다.
결측치가 있는 컬럼은: agent입니다
해당 컬럼에 총 2593개의 결측치가 존재합니다.
결측치가 있는 컬럼은: company입니다
해당 컬럼에 총 17985개의 결측치가 존재합니다.
```

결측치 처리

```
# agent와 company로 예약했는지의 여부
# agent(머 여러종류가 있겠지)로 예약했으면 1, 빈칸이면 0 (compamy도 동일한 방식)
train[['agent','company']] = train[['agent','company']].notna().astype(int)
test[['agent','company']] = test[['agent','company']].notna().astype(int)


# reserved_room_type과 assigned_room_type이 같은지의 여부
train['room_type'] = (train['reserved_room_type'] == train['assigned_room_type']).astype(int)
test['room_type'] = (test['reserved_room_type'] == test['assigned_room_type']).astype(int)

# children의 결측치를 제거
train = train.dropna(subset=['children'], axis=0)
test = test.dropna(subset=['children'], axis=0)
```

원래 reserved_room_type과 assigned_room_type이 같으면 T, 다르면 F를 출력함

but .astype(int) 때문에 같으면 1, 다르면 0 출력

인코딩

```
# one hot encoding
train = pd.get_dummies(train, columns = ['hotel'])
test = pd.get_dummies(test, columns = ['hotel'])

# month 변수 : 날짜를 숫자로 변환
train['arrival_date_month'] = train.arrival_date_month.map({'January' : 1, 'February' : 2, 'March' : 3, 'April' : 4, 'May' : 5,
                                    'June' : 6, 'July' : 7, 'August' : 8, 'September' : 9, 'October' : 10, 'November' : 11, 'Decembe
test['arrival_date_month'] = test.arrival_date_month.map({'January' : 1, 'February' : 2, 'March' : 3, 'April' : 4, 'May' : 5,
                                    'June' : 6, 'July' : 7, 'August' : 8, 'September' : 9, 'October' : 10, 'November' : 11, 'Decembe

# target encoding
encoder = ce.target_encoder.TargetEncoder(cols=['meal', 'distribution_channel', 'customer_type', 'deposit_type'])
encoder.fit(train[['meal', 'distribution_channel', 'customer_type', 'deposit_type']], train['is_canceled']);
train[['meal', 'distribution_channel', 'customer_type', 'deposit_type']] = encoder.transform(train[['meal', 'distribution_channel', 'c
test[['meal', 'distribution_channel', 'customer_type', 'deposit_type']] = encoder.transform(train[['meal', 'distribution_channel', 'cu
```

천저리 완료된 데이터 확인

```
train = train.drop(['country', 'market_segment', 'reserved_room_type', 'assigned_room_type', 'name', 'email', 'phone-number', 'credit_
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 76406 entries, 0 to 76408
Data columns (total 28 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   lead_time                       76406 non-null  int64  
 1   arrival_date_year               76406 non-null  int64  
 2   arrival_date_month              76406 non-null  int64  
 3   arrival_date_week_number        76406 non-null  int64  
 4   arrival_date_day_of_month       76406 non-null  int64  
 5   stays_in_weekend_nights         76406 non-null  int64  
 6   stays_in_week_nights            76406 non-null  int64  
 7   adults                          76406 non-null  int64  
 8   children                        76406 non-null  float64
 9   babies                          76406 non-null  int64  
 10  meal                            76406 non-null  float64
 11  distribution_channel            76406 non-null  float64
 12  is_repeated_guest               76406 non-null  int64  
 13  previous_cancellations          76406 non-null  int64  
 14  previous_bookings_not_canceled  76406 non-null  int64  
 15  booking_changes                 76406 non-null  int64  
 16  deposit_type                    76406 non-null  float64
 17  agent                           76406 non-null  int64  
 18  company                         76406 non-null  int64  
 19  days_in_waiting_list            76406 non-null  int64  
 20  customer_type                   76406 non-null  float64
 21  adr                             76406 non-null  float64
 22  required_car_parking_spaces     76406 non-null  int64  
 23  total_of_special_requests       76406 non-null  int64  
 24  is_canceled                     76406 non-null  int64  
 25  room_type                       76406 non-null  int64  
 26  hotel_City Hotel                76406 non-null  uint8  
 27  hotel_Resort Hotel              76406 non-null  uint8  
dtypes: float64(6), int64(20), uint8(2)
memory usage: 15.9 MB
```

```
test = test.drop(['country', 'market_segment', 'reserved_room_type', 'assigned_room_type','name', 'email', 'phone-number', 'credit_car
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19101 entries, 0 to 19101
Data columns (total 27 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   lead_time                       19101 non-null  int64  
 1   arrival_date_year               19101 non-null  int64  
 2   arrival_date_month              19101 non-null  int64  
 3   arrival_date_week_number        19101 non-null  int64  
 4   arrival_date_day_of_month       19101 non-null  int64  
 5   stays_in_weekend_nights         19101 non-null  int64  
 6   stays_in_week_nights            19101 non-null  int64  
 7   adults                          19101 non-null  int64  
 8   children                        19101 non-null  float64
 9   babies                          19101 non-null  int64  
 10  meal                            19101 non-null  float64
 11  distribution_channel            19101 non-null  float64
 12  is_repeated_guest               19101 non-null  int64  
 13  previous_cancellations          19101 non-null  int64  
 14  previous_bookings_not_canceled  19101 non-null  int64  
 15  booking_changes                 19101 non-null  int64  
 16  deposit_type                    19101 non-null  float64
 17  agent                           19101 non-null  int64  
 18  company                         19101 non-null  int64  
 19  days_in_waiting_list            19101 non-null  int64  
 20  customer_type                   19101 non-null  float64
 21  adr                             19101 non-null  float64
 22  required_car_parking_spaces     19101 non-null  int64  
 23  total_of_special_requests       19101 non-null  int64  
 24  room_type                       19101 non-null  int64  
 25  hotel_City Hotel                19101 non-null  uint8  
 26  hotel_Resort Hotel              19101 non-null  uint8  
dtypes: float64(6), int64(19), uint8(2)
memory usage: 3.8 MB
```

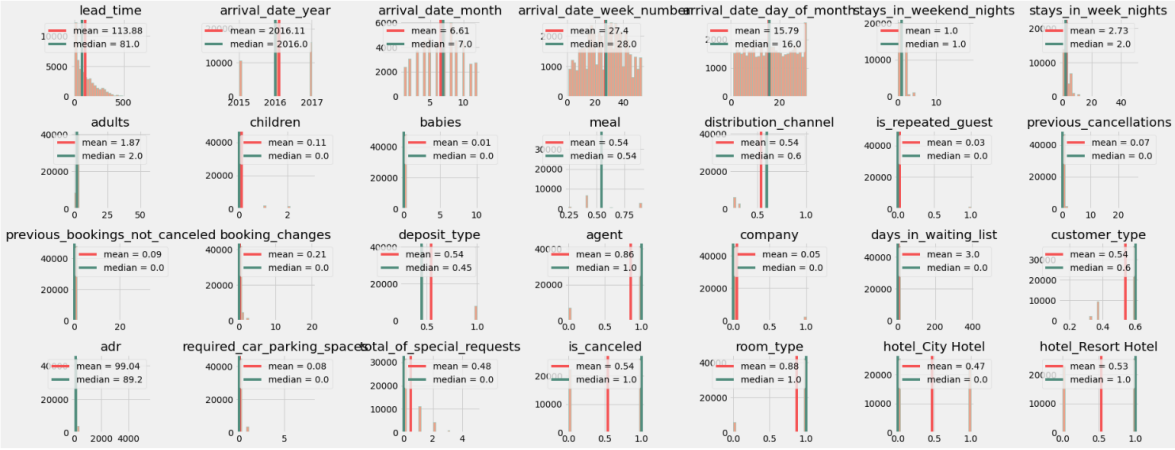
2. EDA

```
data_description = train.describe()
data_description
```

	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_number	arrival_date_day_of_month	stays_in_weekend_nights	stays_in_week_nights	adults	children	babies	...	company	days_in_waiting_list	customer_type	adr	required_car_parking_spaces	total_of_special_requests	is_canceled	room_type	hotel_City Hotel	hotel_Resort Hotel
count	76406.000000	76406.000000	76406.000000	76406.000000	76406.000000	76406.000000	76406.000000	76406.000000	76406.000000	76406.000000	...	76406.000000	76406.000000	76406.000000	76406.000000	76406.000000	76406.000000	76406.000000	76406.000000	76406.000000	76406.000000
mean	103.943997	2016.156781	6.557535	27.188284	15.815080	0.926498	2.501522	1.856032	0.104900	0.008180	...	0.056435	2.344620	0.871113	101.891700	0.062403	0.571866	0.371083	0.875468	0.664738	0.335262
std	106.820421	0.708405	3.098747	13.639586	8.780418	1.002198	1.910022	0.576647	0.399484	0.102829	...	0.230762	17.876717	0.068004	51.930666	0.246232	0.793970	0.483098	0.330190	0.472085	0.472085
min	0.000000	2015.000000	1.000000	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.118980	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	18.000000	2016.000000	4.000000	16.000000	8.000000	0.000000	1.000000	2.000000	0.000000	0.000000	...	0.000000	0.000000	0.408230	69.500000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000
50%	69.000000	2016.000000	7.000000	28.000000	16.000000	1.000000	2.000000	2.000000	0.000000	0.000000	...	0.000000	0.000000	0.408230	94.500000	0.000000	0.000000	0.000000	1.000000	1.000000	0.000000
75%	160.000000	2017.000000	9.000000	38.000000	23.000000	2.000000	3.000000	2.000000	0.000000	0.000000	...	0.000000	0.000000	0.408230	126.000000	0.000000	1.000000	1.000000	1.000000	1.000000	1.000000
max	737.000000	2017.000000	12.000000	53.000000	31.000000	19.000000	50.000000	55.000000	9.000000	10.000000	...	1.000000	391.000000	0.408230	5400.000000	8.000000	5.000000	1.000000	1.000000	1.000000	1.000000

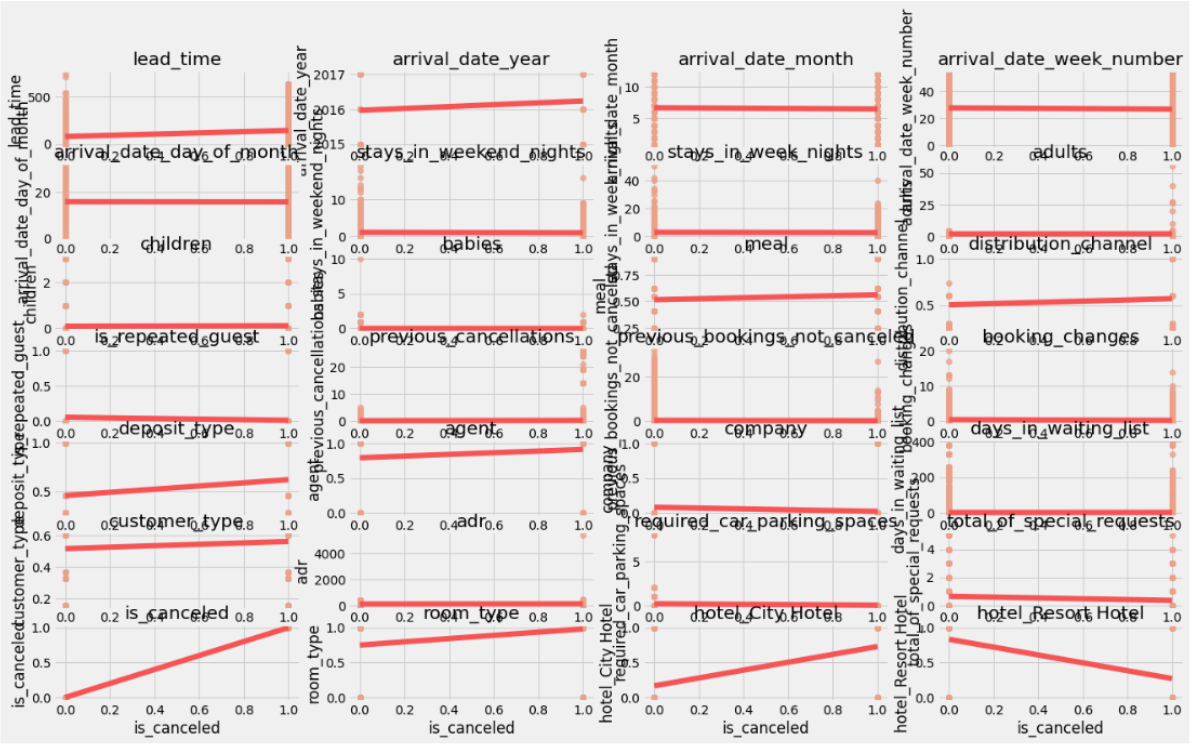
8 rows × 28 columns

```
interest_coloumns = train.columns
plt.style.use('fivethirtyeight')
fig, ax =plt.subplots(4, 7, figsize = (25,10))
column_idx = 0
for i in range(4):
  for j in range(7):
    ax[i][j].hist(train[interest_coloumns[column_idx]], bins=30, color='#eaa18a', edgecolor='#7bcabf')
    ax[i][j].set_title(interest_coloumns[column_idx])
    ax[i][j].axvline(data_description[interest_coloumns[column_idx]]['mean'], c='#f55354', label = f"mean = {round(data_description[in
    ax[i][j].axvline(data_description[interest_coloumns[column_idx]]['50%'], c='#518d7d', label = f"median = {round(data_description[i
    ax[i][j].legend()
    column_idx += 1
fig.tight_layout()
```



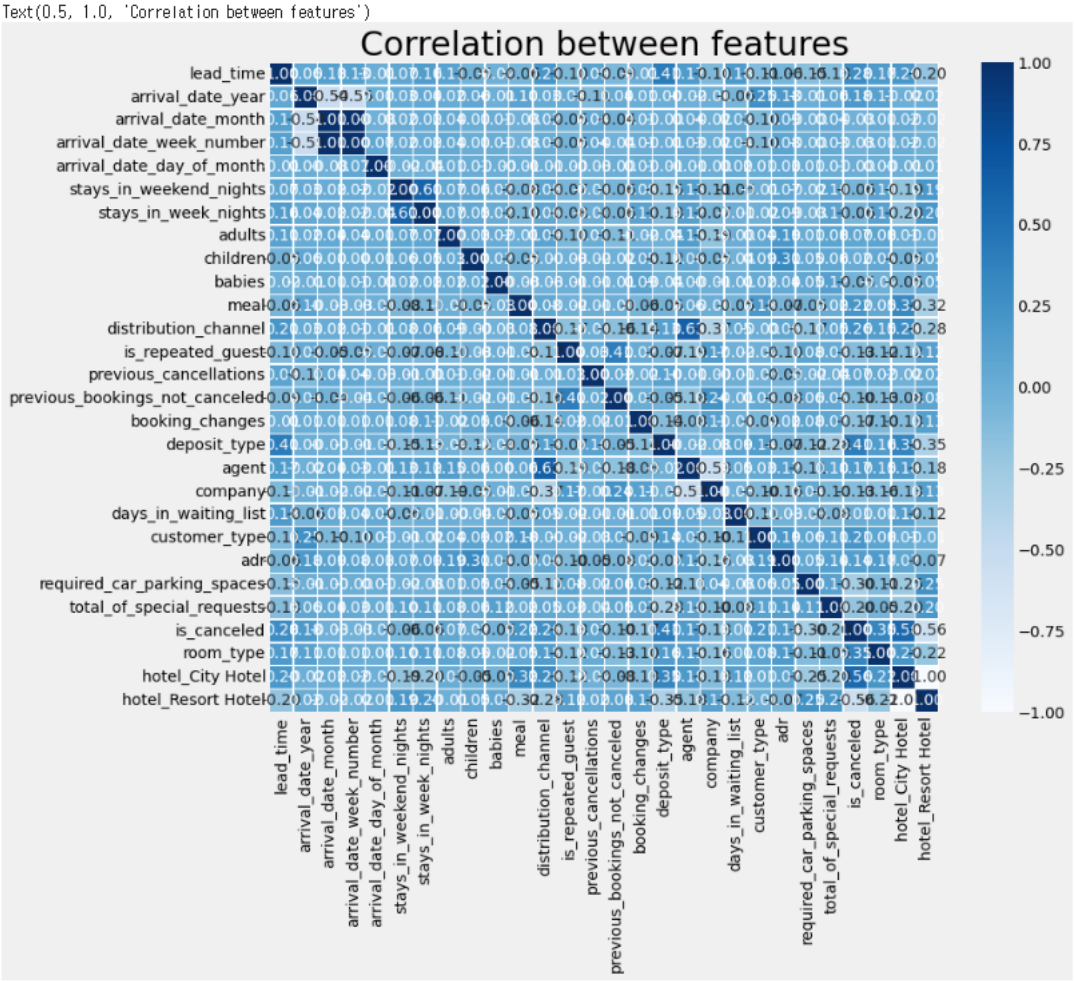

```
def visualize(axx, yfield):
  sns.regplot(x='is_canceled', y=yfield, data=train,  color='#eaa18a', line_kws=  {'color': '#f55354'} , ax = axx)
  axx.set_title(yfield)
```

```
figure, ((ax1,ax2,ax3,ax4),(ax5,ax6,ax7,ax8),(ax9,ax10,ax11,ax12),(ax13,ax14,ax15,ax16),(ax17,ax18,ax19,ax20),(ax21,ax22,ax23,ax24),(a
figure.set_size_inches(20,12)
for i in range(len(interest_coloumns)):
  visualize(eval(f'ax{i+1}'),interest_coloumns[i])
figure.tight_layout()
```




```
plt.figure(figsize=(12,10))
sns.heatmap(data = train.corr(method='pearson'), annot=True, fmt = '.2f', linewidths=.5, cmap='Blues')
plt.title('Correlation between features', fontsize=30)
```

```
Text(0.5, 1.0, 'Correlation between features')
```



3. 변수 선택

```
X = train.drop(["is_canceled"], axis = 1)
y = train['is_canceled']
```

선택한 독립변수 :

'country', 'market_segment', 'reserved_room_type', 'assigned_room_type','name', 'email', 'phone-number', 'credit_card' feature만 제거

제거한 이유 :

'country', 'name', 'email', 'phone-number', 'credit_card' 는 고객의 고유 개인 정보이기 때문에 target변수와 관련 없다고 판단

reserved_room_type', 'assigned_room_type'은 앞에서 room_type 변수로 합쳤음

'market_segment'과 distribution_channel이 유사해 market_segment 삭제

4. Modeling

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = 0.2, random_state = 777)
```

parameter 튜닝

```
# RandomForest parameter 튜닝

rf_clf = RandomForestClassifier(random_state=777)

grid_parameters={'max_depth':[15], 'min_samples_leaf':[2, 3, 4], 'n_estimators':[100, 200, 300]}
grid_rf = GridSearchCV(rf_clf, param_grid = grid_parameters, cv = 5, refit = True)
grid_rf.fit(X_train,y_train)

print('GridSearchCV 최적 파라미터:',grid_rf.best_params_)
print("GridSearchCV 최고 정확도:",grid_rf.best_score_)
```

```
GridSearchCV 최적 파라미터: {'max_depth': 15, 'min_samples_leaf': 2, 'n_estimators': 300}
GridSearchCV 최고 정확도: 0.829821320356749
```

```
# Decision tree parameter 튜닝

dt_clf = DecisionTreeClassifier(random_state=777)

grid_parameters={'max_depth':[13,14,15],'min_samples_leaf':[4,5,6],'min_samples_split':[2,3,4]}
grid_dtree = GridSearchCV(dt_clf, param_grid=grid_parameters, cv = 5, refit = True)
grid_dtree.fit(X_train,y_train)

print('GridSearchCV 최적 파라미터:',grid_dtree.best_params_)
print("GridSearchCV 최고 정확도:",grid_dtree.best_score_)
```

```
GridSearchCV 최적 파라미터: {'max_depth': 15, 'min_samples_leaf': 4, 'min_samples_split': 2}
GridSearchCV 최고 정확도: 0.8141809789184039
```

```
# GB parameter 튜닝

gb = GradientBoostingClassifier(random_state=777)

grid_parameters={'max_depth':[4,5],'learning_rate':[0.3,0.4],'min_samples_leaf':[3,4]}
grid_gb = GridSearchCV(gb, param_grid=grid_parameters, cv = 5, refit = True)
grid_gb.fit(X_train,y_train)

print('GridSearchCV 최적 파라미터:',grid_gb.best_params_)
print("GridSearchCV 최고 정확도:",grid_gb.best_score_)
```

```
GridSearchCV 최적 파라미터: {'learning_rate': 0.4, 'max_depth': 5, 'min_samples_leaf': 4}
GridSearchCV 최고 정확도: 0.8347948311362793
```

```
# XGB parameter 튜닝

xgb = XGBClassifier(random_state=777)

grid_parameters={'max_depth':[4,5],'early_stopping_rounds':[100,200],'learning_rate' :[0.3,0.4]}
grid_xgb = GridSearchCV(xgb, param_grid=grid_parameters, cv = 5, refit = True)
grid_xgb.fit(X_train,y_train)

print('GridSearchCV 최적 파라미터:',grid_xgb.best_params_)
print("GridSearchCV 최고 정확도:",grid_xgb.best_score_)
```

```
GridSearchCV 최적 파라미터: {'early_stopping_rounds': 100, 'learning_rate': 0.4, 'max_depth': 5}
GridSearchCV 최고 정확도: 0.8355310451208616
```

```
# Adaboost parameter 튜닝

ada = AdaBoostClassifier(random_state=777)
```

```
grid_parameters={'n_estimators':[100,200],'learning_rate':[0.3,0.4]}
grid_ada = GridSearchCV(ada, param_grid=grid_parameters, cv = 5, refit = True)
grid_ada.fit(X_train,y_train)

print('GridSearchCV 최적 파라미터:',grid_ada.best_params_)
print("GridSearchCV 최고 정확도:",grid_ada.best_score_)
```

```
GridSearchCV 최적 파라미터: {'learning_rate': 0.4, 'n_estimators': 200}
GridSearchCV 최고 정확도: 0.8074896627342719
```

```
# LogisticRegression parameter 튜닝

lgr = LogisticRegression(random_state=777)

grid_parameters = {'penalty':['l2', 'l1'],'C':[0.01, 0.1, 1]}
grid_lgr = GridSearchCV(lgr, param_grid = grid_parameters, cv = 5, refit = True)
grid_lgr.fit(X_train, y_train)

print('GridSearchCV 최적 파라미터:',grid_lgr.best_params_)
print("GridSearchCV 최고 정확도:",grid_lgr.best_score_)
```

```
GridSearchCV 최적 파라미터: {'C': 1, 'penalty': 'l2'}
GridSearchCV 최고 정확도: 0.7436847363304168
```

voting

```
# Soft Voting

rf_clf = RandomForestClassifier(max_depth= 12, min_samples_leaf=3, n_estimators=150,random_state=42)
dt_clf = DecisionTreeClassifier(max_depth= 12, min_samples_leaf= 5, min_samples_split=2,random_state=42)
gb = GradientBoostingClassifier(learning_rate = 0.3, max_depth=5, min_samples_leaf=4,random_state=42)
xgb = XGBClassifier(early_stopping_rounds = 100, learning_rate = 0.4, max_depth= 5,random_state=42)
ada = AdaBoostClassifier(learning_rate = 0.4, n_estimators = 200, random_state=42)
lgr = LogisticRegression(penalty='l2', C=1 , random_state=42)

vo_clf=VotingClassifier(estimators=[('RF',rf_clf),('DT', dt_clf),('GB',gb),('XGB',xgb),('ADA',ada),('LGR',lgr)], voting='soft', random

vo_clf.fit(X_train,y_train)
pred = vo_clf.predict(X_test)
print( "Soft Voting 정확도:",accuracy_score(y_test,pred))
```

```
Soft Voting 정확도: 0.8415796044418161
```

fin

vo_clf(우리가 앞에서 만든 model인)로 실제 임의의 데이터 test를 넣어 취소 여부를 직접 예측하는 코드

```
test = pd.read_csv('test.csv')
final_pred = vo_clf.predict(test)
final_pred.to_csv("team_C.csv") # 예측한 파일을 csv로 저장
```

project comment

결과적으로 예측한 취소 여부의 정확도는 다른 팀에 비해 낮은 수준이었다. 팀별로 사용했던 model은 비슷했기 때문에 modeling과정은 다들 비슷했다. 전처리와 변수 선택 과정에 있어서 옳지 않은 판단을 한 것 같다. 결측치가 많은 변수에서 결측치를 0으로 대체했던 것과 여러 인코딩 방법중 target 인코딩을 사용했던 것이 문제인 것 같다. 모델의 성능만을 높이는 데에만 집중하다보니 전처리 과정을 놓친것

같다. 이 데이터는 특히 변수가 많기 때문에 변수 선택에 있어서 target 변수간의 중요도를 잘못판단 했을 수 있다는 생각을 했다. 변수 중요도를 EDA를 통해 확인했지만 다중공선성의 문제도 생각해봐야한다.

내가 생각한 해결 방법 : 변수 줄이기, 다른 인코딩 방법 사용해보기, 결측치를 0말고 데이터를 더 보존할 수 있는 값으로 대체 해보기,...

우리 팀이 진행한 방법

- **결측치 찾고 해결방안 생각하기**
- **인코딩 방안 생각하기**
- **타겟변수의 분포 확인하기**
- **개별 변수들 사이의 상관관계 파악하기**
- **개별 변수들의 분포 확인하기**
- **개별 변수들과 타겟 변수사이의 관계 파악하기**
- **존재하는 변수들로 새로운 변수 만들기**