# 《商务大数据分析与决策》期末课程报告

姓名：曾诚
学号：41951020
专业：市场营销（金融服务与营销）

## 导入需要的库

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import folium
#import eli5 # Feature importance evaluation
# 机器学习
import sklearn
from sklearn.model_selection import train_test_split, KFold, cross_validate, cross_
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import LabelEncoder,OneHotEncoder,StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import GridSearchCV
from sklearn.svm import LinearSVC
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression, SGDClassifier, RidgeClassifier
from sklearn.model_selection import cross_validate,train_test_split
from sklearn.pipeline import make_pipeline
from sklearn.datasets import make_classification


from sklearn import metrics
from sklearn.metrics import plot_roc_curve
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import plot_precision_recall_curve
import matplotlib.pyplot as plt
from sklearn.metrics import average_precision_score

# Other Libraries
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline
from imblearn.pipeline import make_pipeline as imbalanced_make_pipeline
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import NearMiss
from imblearn.metrics import classification_report_imbalanced
from sklearn.metrics import precision_score, recall_score, f1_score, roc_auc_score,
#from collections import Counter
#from sklearn.model_selection import KFold, StratifiedKFold
#import warnings
#warnings.filterwarnings("ignore")
```

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

# 一、数据预处理

### 1.1导入数据

In [ ]:
```
data_origin = pd.read_csv('hotel_bookings.csv')
#这里的代码路径根据用户保存路径有所不同
data_origin.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 119390 entries, 0 to 119389
Data columns (total 32 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   hotel                           119390 non-null  object
 1   is_canceled                     119390 non-null  int64
 2   lead_time                       119390 non-null  int64
 3   arrival_date_year               119390 non-null  int64
 4   arrival_date_month              119390 non-null  object
 5   arrival_date_week_number        119390 non-null  int64
 6   arrival_date_day_of_month       119390 non-null  int64
 7   stays_in_weekend_nights         119390 non-null  int64
 8   stays_in_week_nights            119390 non-null  int64
 9   adults                          119390 non-null  int64
 10  children                        119386 non-null  float64
 11  babies                          119390 non-null  int64
 12  meal                            119390 non-null  object
 13  country                         118902 non-null  object
 14  market_segment                  119390 non-null  object
 15  distribution_channel            119390 non-null  object
 16  is_repeated_guest               119390 non-null  int64
 17  previous_cancellations          119390 non-null  int64
 18  previous_bookings_not_canceled  119390 non-null  int64
 19  reserved_room_type              119390 non-null  object
 20  assigned_room_type              119390 non-null  object
 21  booking_changes                 119390 non-null  int64
 22  deposit_type                    119390 non-null  object
 23  agent                           103050 non-null  float64
 24  company                         6797 non-null    float64
 25  days_in_waiting_list            119390 non-null  int64
 26  customer_type                   119390 non-null  object
 27  adr                             119390 non-null  float64
 28  required_car_parking_spaces     119390 non-null  int64
 29  total_of_special_requests       119390 non-null  int64
 30  reservation_status              119390 non-null  object
 31  reservation_status_date         119390 non-null  object
dtypes: float64(4), int64(16), object(12)
memory usage: 29.1+ MB
```

总结一下数据列的基本情况 数据主要包含了以下三个方面的信息：

- 订单信息：预订相关时间/状态信息
- 房间信息：酒店房间的价格/类型特征
- 客户信息：客户自身的相关信息

### 1.2 数据预处理

In [ ]:
```
data = data_origin.copy()
missing=data.isnull().sum(axis=0)
missing[missing!=0]
```

Out[ ]:
```
children         4
country        488
agent        16340
company     112593
dtype: int64
```

In [ ]:
```
#缺失值处理
data.children.fillna(data.children.mode()[0],inplace=True)
data.country.fillna(data.country.mode()[0],inplace=True)
data.agent.fillna(0, inplace=True)
data.drop('company', axis=1,inplace=True)
#(执行一遍就可以了)
```

In [ ]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 119390 entries, 0 to 119389
Data columns (total 31 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   hotel                           119390 non-null  object
 1   is_canceled                     119390 non-null  int64
 2   lead_time                       119390 non-null  int64
 3   arrival_date_year               119390 non-null  int64
 4   arrival_date_month              119390 non-null  object
 5   arrival_date_week_number        119390 non-null  int64
 6   arrival_date_day_of_month       119390 non-null  int64
 7   stays_in_weekend_nights         119390 non-null  int64
 8   stays_in_week_nights            119390 non-null  int64
 9   adults                          119390 non-null  int64
 10  children                        119390 non-null  float64
 11  babies                          119390 non-null  int64
 12  meal                            119390 non-null  object
 13  country                         119390 non-null  object
 14  market_segment                  119390 non-null  object
 15  distribution_channel            119390 non-null  object
 16  is_repeated_guest               119390 non-null  int64
 17  previous_cancellations          119390 non-null  int64
 18  previous_bookings_not_canceled  119390 non-null  int64
 19  reserved_room_type              119390 non-null  object
 20  assigned_room_type              119390 non-null  object
 21  booking_changes                 119390 non-null  int64
 22  deposit_type                    119390 non-null  object
 23  agent                           119390 non-null  float64
 24  days_in_waiting_list            119390 non-null  int64
 25  customer_type                   119390 non-null  object
 26  adr                             119390 non-null  float64
 27  required_car_parking_spaces     119390 non-null  int64
 28  total_of_special_requests       119390 non-null  int64
 29  reservation_status              119390 non-null  object
 30  reservation_status_date         119390 non-null  object
dtypes: float64(3), int64(16), object(12)
memory usage: 28.2+ MB
```

### 1.3异常值处理

In [ ]:
```
zero_guest=data[data[['adults', 'children', 'babies']].sum(axis=1) == 0]
```

```
data.drop(zero_guest.index, inplace=True)#筛选入住总人数为0的数据

zero_days = data[data[['stays_in_weekend_nights','stays_in_week_nights']].sum(axis=1
data.drop(zero_days.index, inplace=True)#筛选入住总天数为0的数据

data.meal.replace("Undefined", "SC", inplace=True)# 餐食类型Undefined/SC合并
```

In [ ]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 118565 entries, 2 to 119389
Data columns (total 31 columns):
 #   Column                          Non-Null Count   Dtype  
---  ------                          --------------   -----  
 0   hotel                           118565 non-null  object 
 1   is_canceled                     118565 non-null  int64  
 2   lead_time                       118565 non-null  int64  
 3   arrival_date_year               118565 non-null  int64  
 4   arrival_date_month              118565 non-null  object 
 5   arrival_date_week_number        118565 non-null  int64  
 6   arrival_date_day_of_month       118565 non-null  int64  
 7   stays_in_weekend_nights         118565 non-null  int64  
 8   stays_in_week_nights            118565 non-null  int64  
 9   adults                          118565 non-null  int64  
 10  children                        118565 non-null  float64
 11  babies                          118565 non-null  int64  
 12  meal                            118565 non-null  object 
 13  country                         118565 non-null  object 
 14  market_segment                  118565 non-null  object 
 15  distribution_channel            118565 non-null  object 
 16  is_repeated_guest               118565 non-null  int64  
 17  previous_cancellations          118565 non-null  int64  
 18  previous_bookings_not_canceled  118565 non-null  int64  
 19  reserved_room_type              118565 non-null  object 
 20  assigned_room_type              118565 non-null  object 
 21  booking_changes                 118565 non-null  int64  
 22  deposit_type                    118565 non-null  object 
 23  agent                           118565 non-null  float64
 24  days_in_waiting_list            118565 non-null  int64  
 25  customer_type                   118565 non-null  object 
 26  adr                             118565 non-null  float64
 27  required_car_parking_spaces     118565 non-null  int64  
 28  total_of_special_requests       118565 non-null  int64  
 29  reservation_status              118565 non-null  object 
 30  reservation_status_date         118565 non-null  object 
dtypes: float64(3), int64(16), object(12)
memory usage: 28.9+ MB
```

In [ ]:
```
data.shape
```

Out[ ]:
```
(118565, 31)
```

到这里，数据的**预处理**工作完成，数据集大小清洗为118565*31

# 二、数据可视化分析

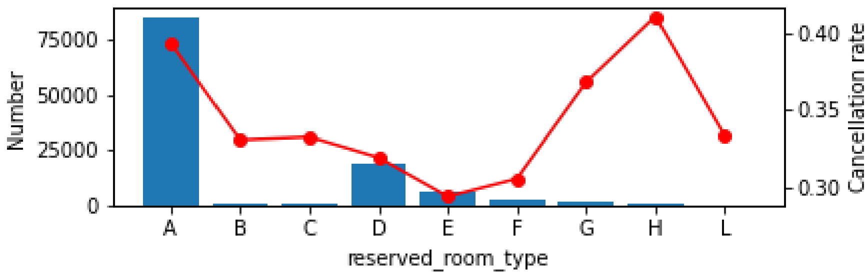
### 2.1 客房信息分析

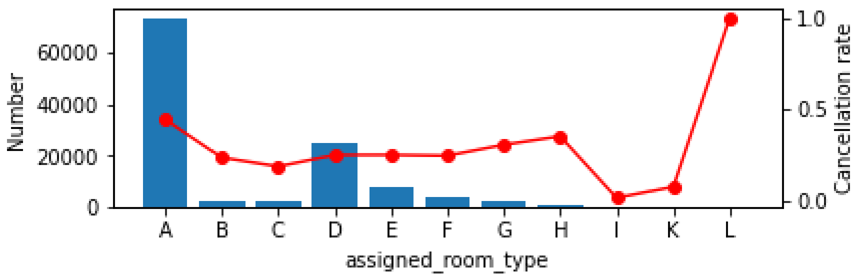
①酒店类型

In [ ]:
```
sns.countplot(x='hotel', hue='is_canceled', data=data)
plt.show()
```


②客房类型

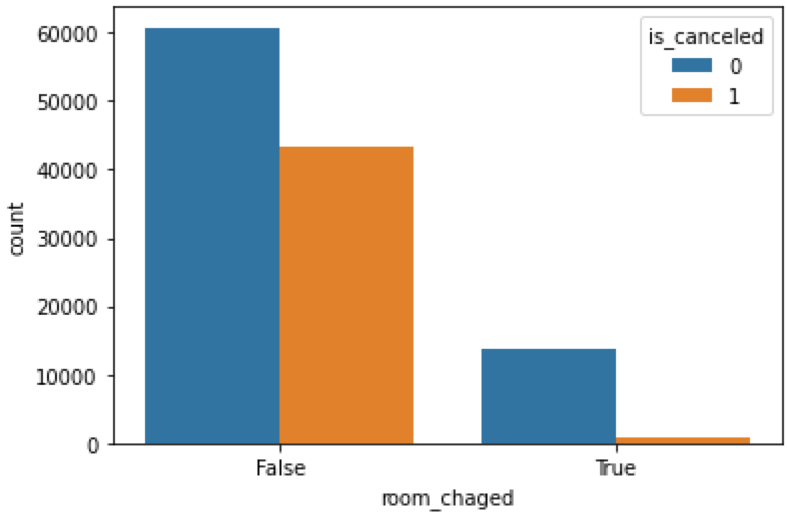
In [ ]:
```
# 查看房间类型与取消预订的关系
index = 1
for room_type in ['reserved_room_type', 'assigned_room_type']:
    # plt.figure(figsize=(6,8))
    ax1 = plt.subplot(2, 1, index)
    index += 1
    ax2 = ax1.twinx()
    ax1.bar(
        data.groupby(room_type).size().index,
        data.groupby(room_type).size())
    ax1.set_xlabel(room_type)
    ax1.set_ylabel('Number')
    ax2.plot(
        data.groupby(room_type)['is_canceled'].mean(), 'ro-')
    ax2.set_ylabel('Cancellation rate')
    plt.show()
```




In [ ]:
```
# 房间类型变更对取消预定的影响
data['room_chaged']=data['reserved_room_type']!=data['assigned_room_type']
sns.countplot(x='room_chaged',hue='is_canceled',data=data)
```

Out[ ]: <AxesSubplot:xlabel='room_chaged', ylabel='count'>

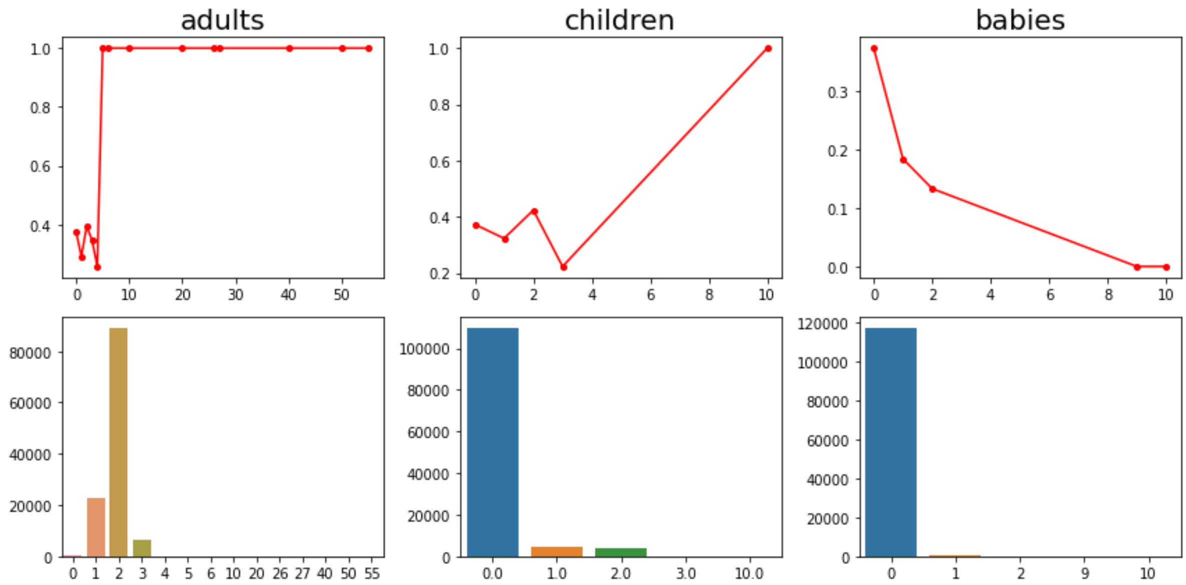



### 2.2 客户信息分析

①入住人数

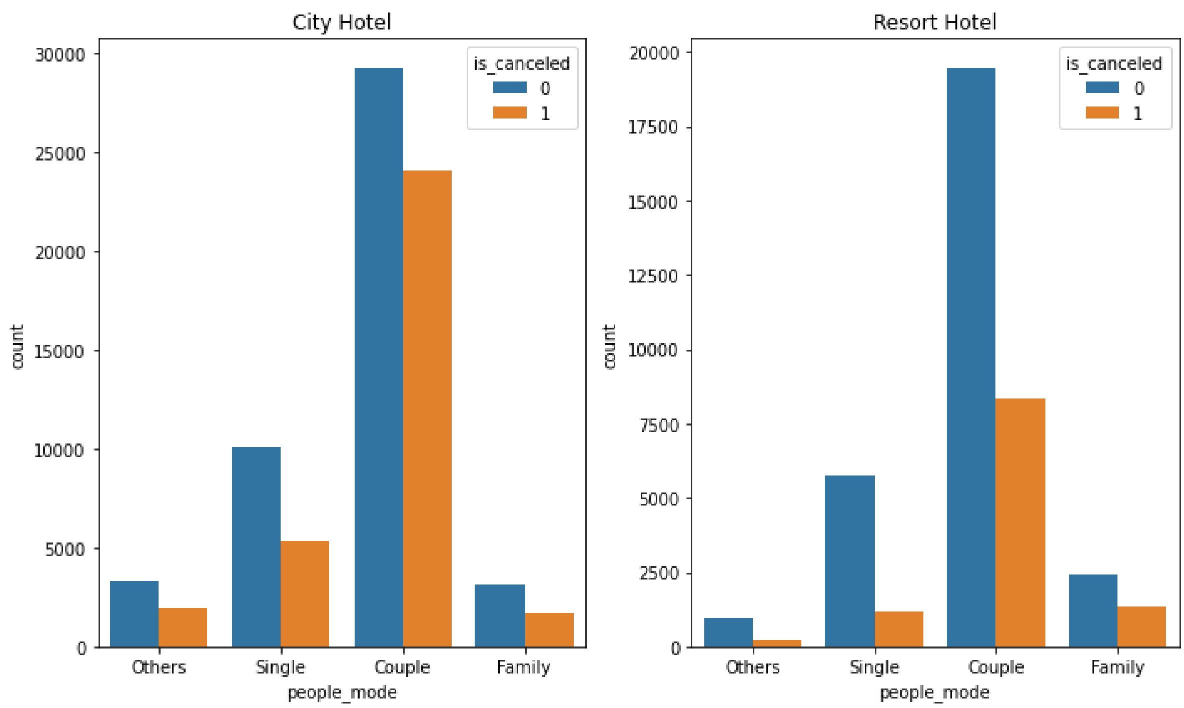
In [ ]:
```
# 查看预定人数与取消预定的关系
plt.figure(figsize=(12, 6))
index = 0
for people in ['adults', 'children', 'babies']:
    index += 1
    plt.subplot(2, 3, index)
    plt.plot(data.groupby(people)['is_canceled'].mean(),
             'ro-',
             ms=4)
    plt.title(people, fontsize=20)
    plt.subplot(2, 3, index + 3)
    people_stats = data[people].value_counts();
    sns.barplot(people_stats.index, people_stats.values)
plt.tight_layout()
plt.show()
```

```
d:\anaconda\lib\site-packages\seaborn\_decorators.py:43: FutureWarning: Pass the fol
lowing variables as keyword args: x, y. From version 0.12, the only valid positional
argument will be `data`, and passing other arguments without an explicit keyword wil
l result in an error or misinterpretation.
  FutureWarning
d:\anaconda\lib\site-packages\seaborn\_decorators.py:43: FutureWarning: Pass the fol
lowing variables as keyword args: x, y. From version 0.12, the only valid positional
argument will be `data`, and passing other arguments without an explicit keyword wil
l result in an error or misinterpretation.
  FutureWarning
d:\anaconda\lib\site-packages\seaborn\_decorators.py:43: FutureWarning: Pass the fol
lowing variables as keyword args: x, y. From version 0.12, the only valid positional
argument will be `data`, and passing other arguments without an explicit keyword wil
l result in an error or misinterpretation.
  FutureWarning
```




In [ ]:

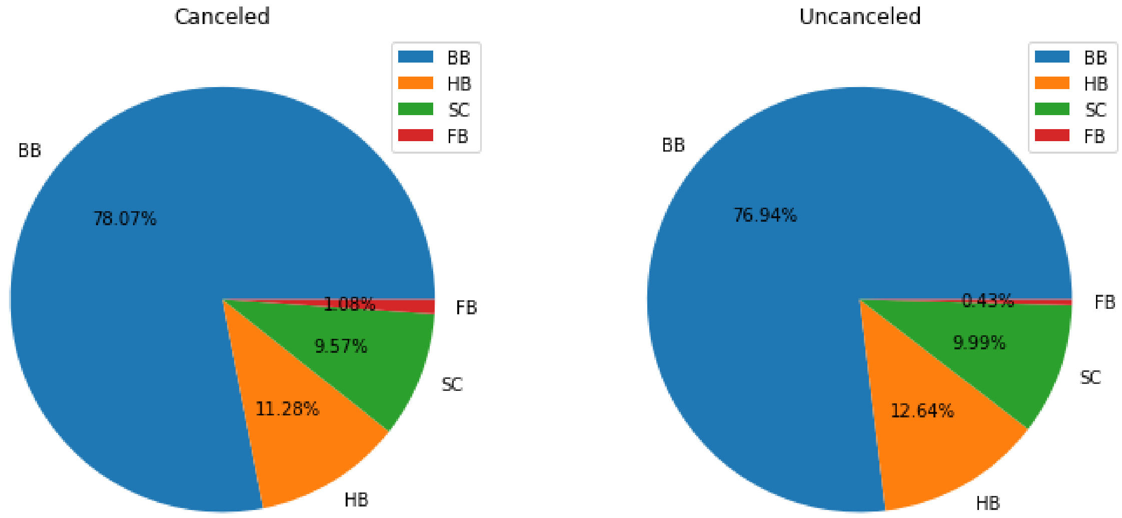
```
# 入住人数模式分析
# 单人
single = (data.adults == 1) & (data.children == 0) & (data.babies == 0)
# 双人
couple = (data.adults == 2) & (data.children == 0) & (data.babies == 0)
# 家庭
family = (data.adults >= 2) & (data.children > 0) | (data.babies > 0)

data['people_mode'] = single.astype(int) + couple.astype(int) * 2 + family.astype(i
plt.figure(figsize=(10,6))
index=1
for hotel_kind in ['City Hotel','Resort Hotel']:
    plt.subplot(1,2,index)
    index+=1
    sns.countplot(x='people_mode',
                  hue='is_canceled',
                  data=data[data.hotel == hotel_kind])
    plt.xticks([0, 1, 2, 3], ['Others', 'Single', 'Couple', 'Family'])
    plt.title(hotel_kind)
plt.tight_layout()
plt.show()
```



②餐食类型

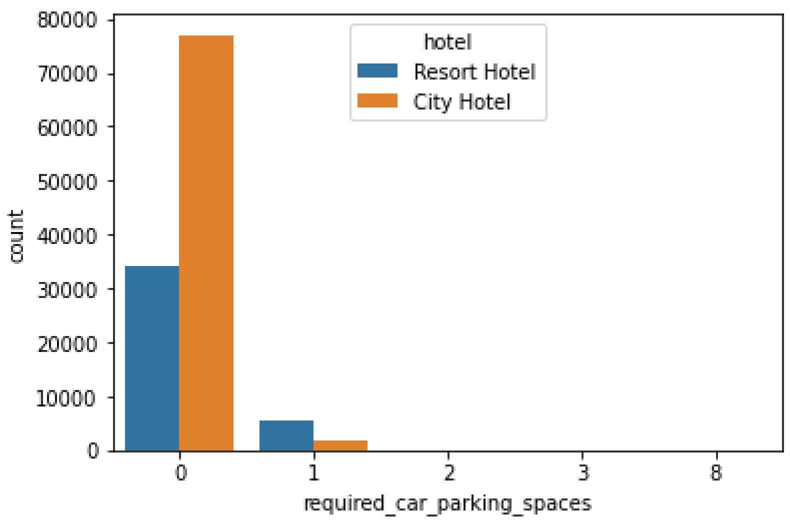
In [ ]:
```
# 查看餐食类型与取消预订的关系
plt.figure(figsize=(12, 6))
plt.subplot(121)
plt.pie(data[data['is_canceled'] == 1].meal.value_counts(),
        labels=data[data['is_canceled'] == 1].meal.value_counts().index,
        autopct="%.2f%%")
plt.legend(loc=1)
plt.title('Canceled')
plt.subplot(122)
plt.pie(data[data['is_canceled'] == 0].meal.value_counts(),
        labels=data[data['is_canceled'] == 0].meal.value_counts().index,
        autopct="%.2f%%")
plt.legend(loc=1)
plt.title('Uncanceled')
```

Out[ ]:
```
Text(0.5, 1.0, 'Uncanceled')
```




③车位需求

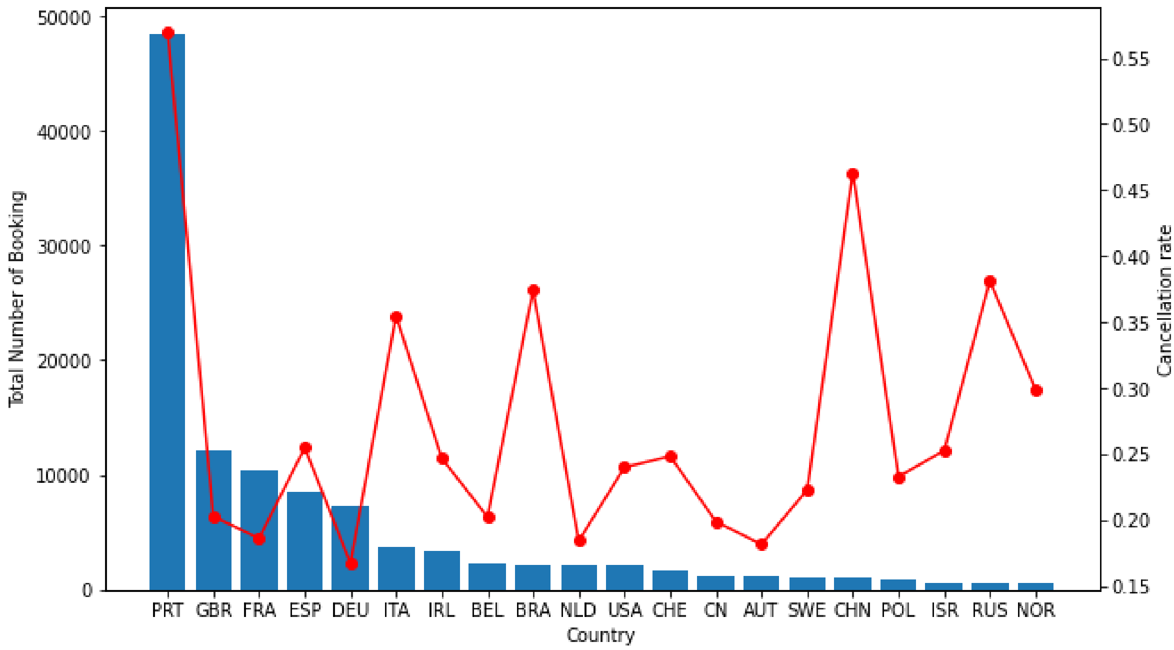
In [ ]:
```
# 车位需求统计
sns.countplot(x='required_car_parking_spaces',hue='hotel',data=data);
```




④国家/地区

In [ ]:
```
# 查看不同国家订单取消率
# 选取预定数前20的国家/地区
countries_20 = list(
    data.groupby('country').size().sort_values(ascending=False).head(20).index)
data[data.country.isin(countries_20)].shape[0] / data.shape[0]

fig, ax1 = plt.subplots(figsize=(10, 6))
ax2 = ax1.twinx()
plt.xticks(range(20), countries_20)
ax1.bar(
    range(20), data[data.country.isin(countries_20)].groupby('country').size().sort_
ax1.set_xlabel('Country')
ax1.set_ylabel('Total Number of Booking')
ax2.plot(
    range(20),
    data[data.country.isin(countries_20)].groupby('country')['is_canceled'].mean().
ax2.set_ylabel('Cancellation rate')
```

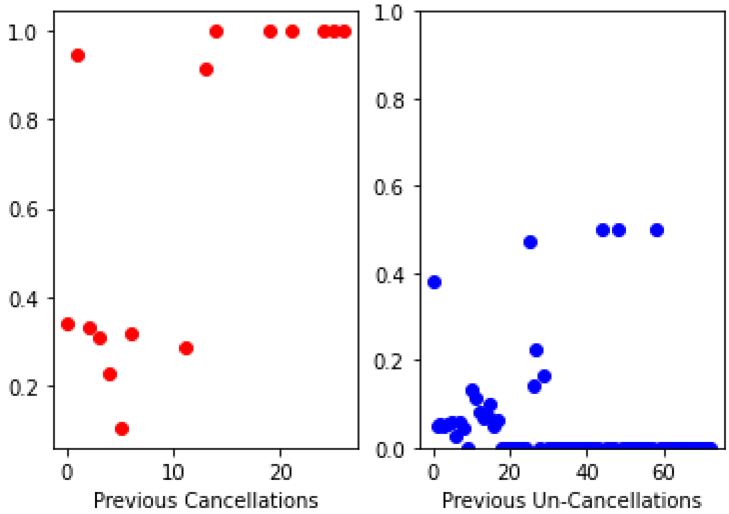
Out[ ]: Text(0, 0.5, 'Cancellation rate')




⑤客户预定历史

In [ ]:
```
# 查看客户预定历史与取消订单的关系
# 是否回头客
tick_label = ['New Guest', 'Repeated Guest']
sns.countplot(x='is_repeated_guest', hue='is_canceled', data=data)
plt.xticks([0, 1], tick_label)

# 之前取消预定次数
plt.subplot(121)
plt.plot(data.groupby('previous_cancellations')['is_canceled'].mean(),
         'ro')
plt.xlabel('Previous Cancellations')
# 之前未取消预定次数
plt.subplot(122)
plt.plot(data.groupby('previous_bookings_not_canceled')['is_canceled'].mean(),
         'bo')
plt.ylim(0, 1)
plt.xlabel('Previous Un-Cancellations')
```

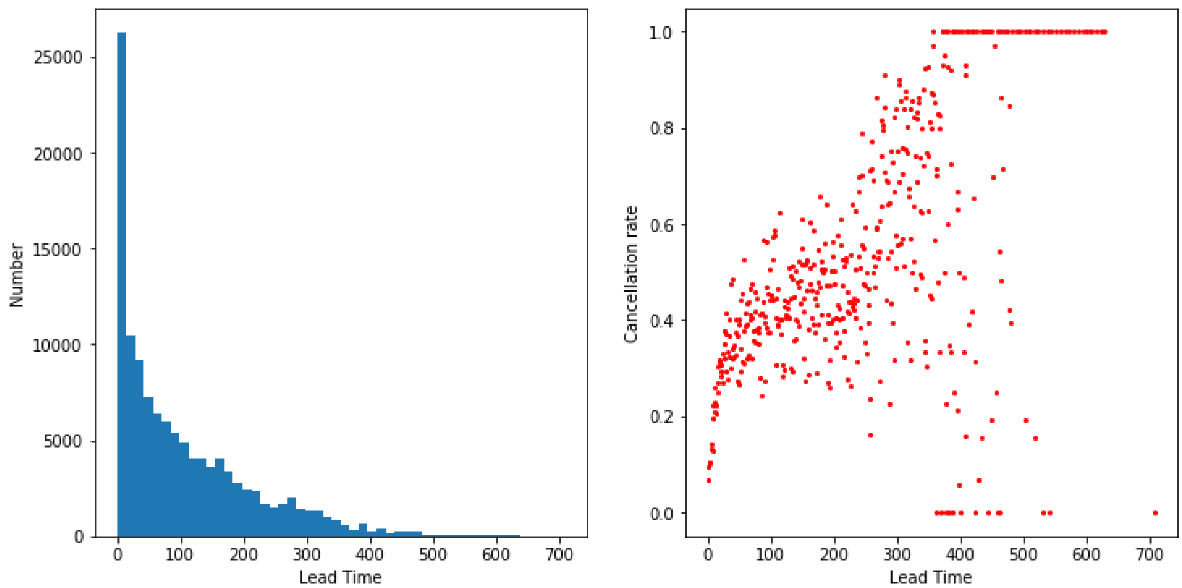
Out[ ]: Text(0.5, 0, 'Previous Un-Cancellations')




2.3 订单信息分析

①提前预定时长

In [ ]:
```
# 提前预定时长的分布情况
plt.figure(figsize=(12, 6))
plt.subplot(121)
plt.hist(data['lead_time'], bins=50)
plt.xlabel('Lead Time')
plt.ylabel('Number')
# 提前预定时长对取消的影响
plt.subplot(122)
plt.plot(data.groupby('lead_time')['is_canceled'].mean().index,
         data.groupby('lead_time')['is_canceled'].mean(),
         'ro',
         markersize=2)
plt.xlabel('Lead Time')
plt.ylabel('Cancellation rate');
```

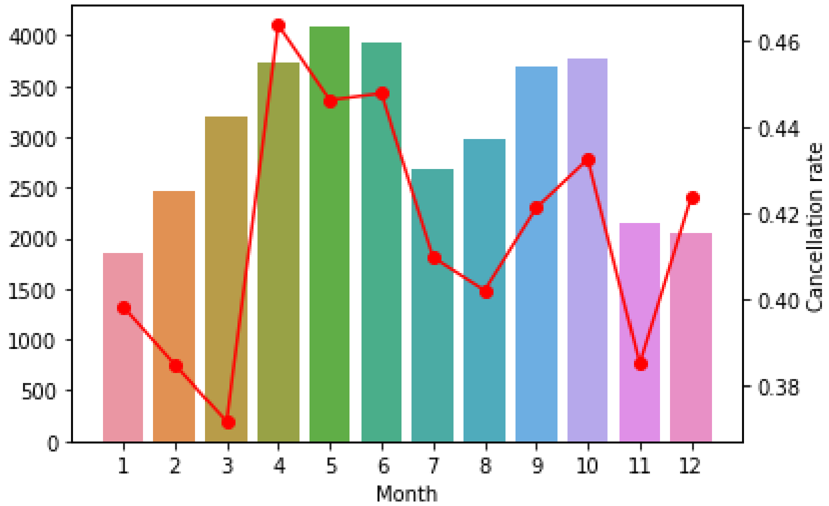


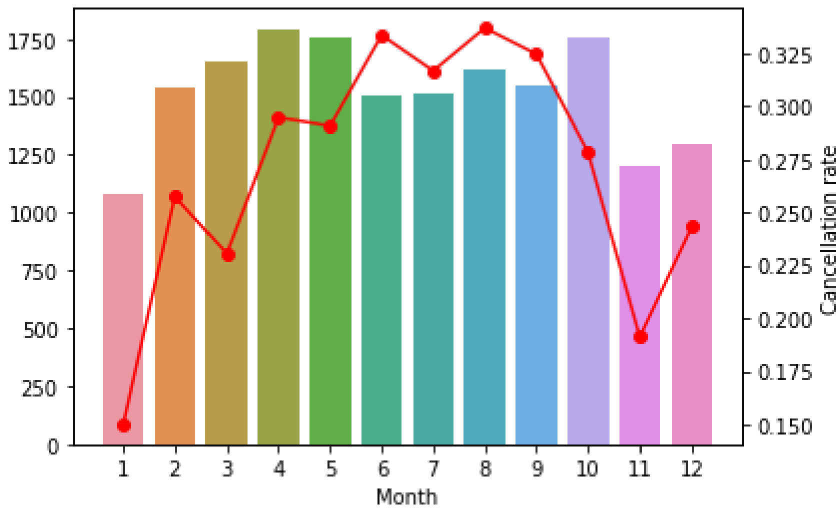

②入住时间

In [ ]:
```
# 不同月份预定和取消情况
ordered_months = [
```

```
    "January", "February", "March", "April", "May", "June", "July", "August",
    "September", "October", "November", "December"
]

for hotel in ['City Hotel','Resort Hotel']:
    fig, ax1 = plt.subplots()
    ax2 = ax1.twinx()
    data_hotel=data[data.hotel==hotel]
    monthly = data_hotel.groupby('arrival_date_month').size()
    monthly /= 2
    monthly.loc[['July', 'August']] = monthly.loc[['July', 'August']] * 2 / 3
    sns.barplot(list(range(1, 13)), monthly[ordered_months], ax=ax1)
    ax2.plot(
    range(12), data_hotel.groupby('arrival_date_month')
    ['is_canceled'].mean()[ordered_months].values, 'ro-')
    ax1.set_xlabel('Month');
    ax2.set_ylabel('Cancellation rate');
```

```
d:\anaconda\lib\site-packages\seaborn\_decorators.py:43: FutureWarning: Pass the fol
lowing variables as keyword args: x, y. From version 0.12, the only valid positional
argument will be `data`, and passing other arguments without an explicit keyword wil
l result in an error or misinterpretation.
  FutureWarning
d:\anaconda\lib\site-packages\seaborn\_decorators.py:43: FutureWarning: Pass the fol
lowing variables as keyword args: x, y. From version 0.12, the only valid positional
argument will be `data`, and passing other arguments without an explicit keyword wil
l result in an error or misinterpretation.
  FutureWarning
```

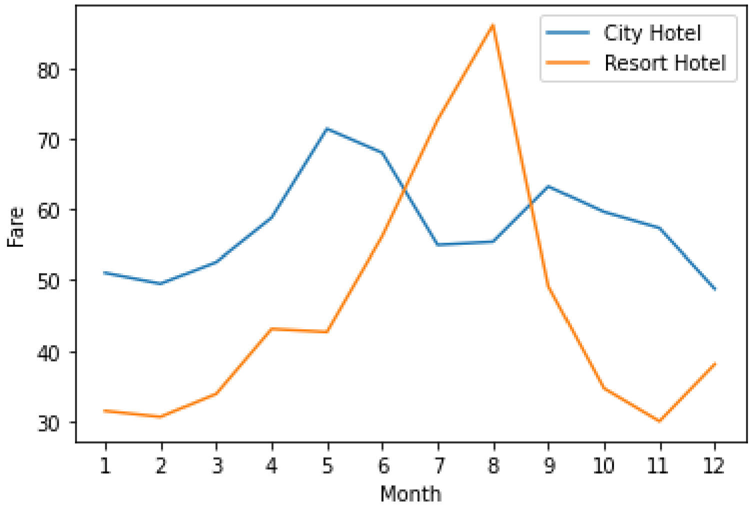



下面结合酒店人均价格波动进行分析

In [ ]:

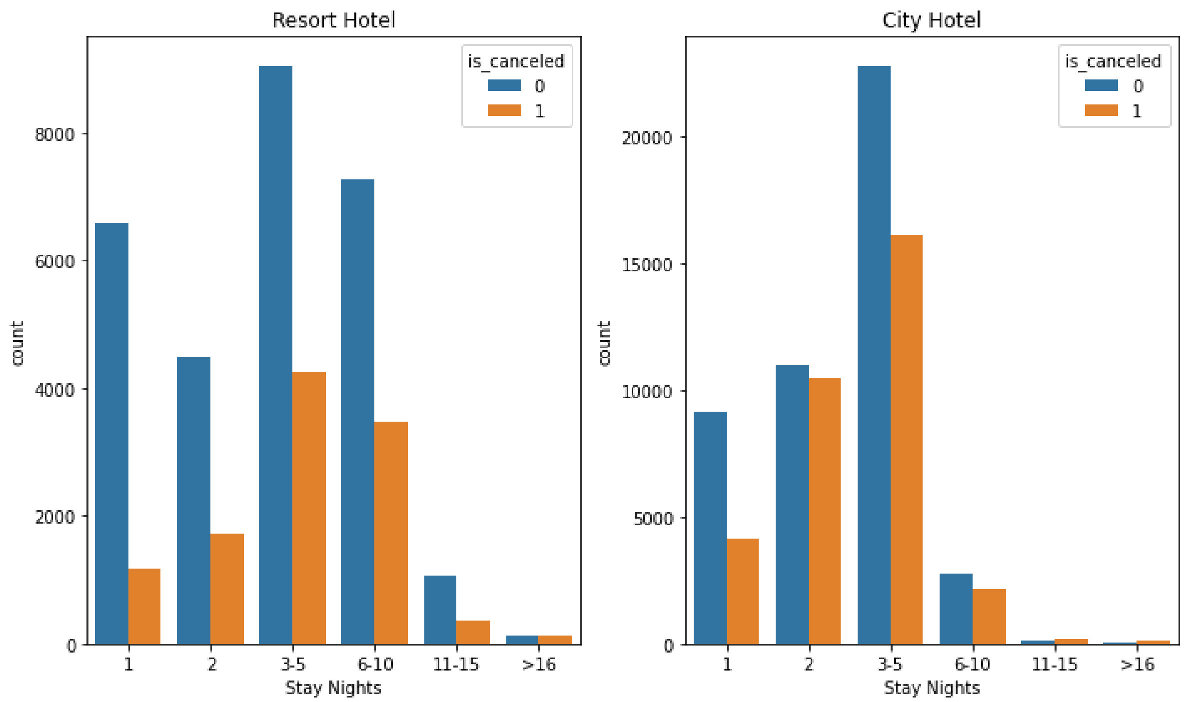
```
# 不同酒店人均价格波动
# 人均价格(不考虑babies)
data['adr_per_person'] = data['adr'] / (data['adults'] + data['children'])
plt.plot(data[data.hotel == 'City Hotel'].groupby('arrival_date_month')
         ['adr_per_person'].mean()[ordered_months],
         label='City Hotel')
plt.plot(data[data.hotel == 'Resort Hotel'].groupby('arrival_date_month')
         ['adr_per_person'].mean()[ordered_months],
         label='Resort Hotel')
plt.xlabel('Month')
plt.ylabel('Fare')
plt.xticks(np.arange(12), range(1, 13))
plt.legend();
```




③入住时长

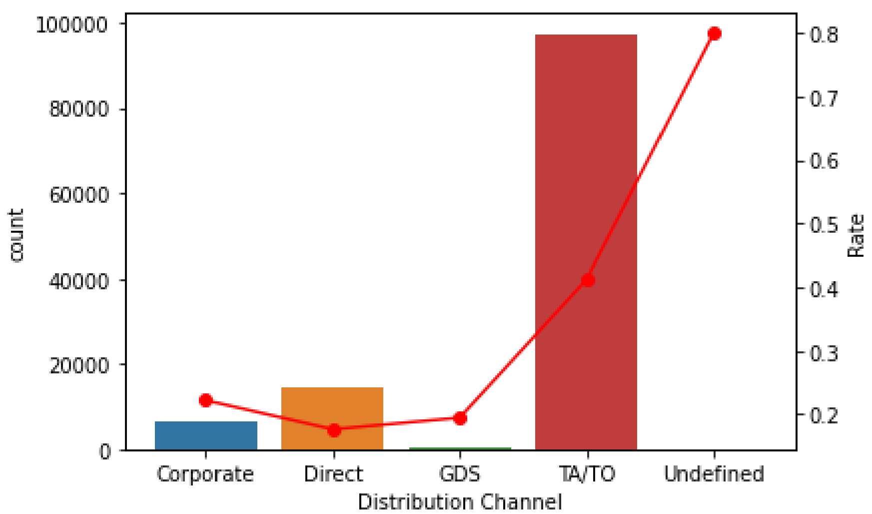
In [ ]:

```
# 预定入住时长对取消预定的影响
data['stay_nights'] = data['stays_in_weekend_nights'] + data['stays_in_week_nights']
# 分布过散，进行数据分桶
bin = [0, 1, 2, 5, 10, 15, np.inf]
data['stay_nights_bin'] = pd.cut(data['stay_nights'], bin,
                                 labels=['1', '2', '3-5', '6-10', '11-15', '>16'])
plt.figure(figsize=(10,6))
plt.subplot(121)
sns.countplot(x='stay_nights_bin', hue='is_canceled',
              data=data[data['hotel'] == 'Resort Hotel'])
plt.xlabel('Stay Nights')
plt.title('Resort Hotel')
plt.subplot(122)
sns.countplot(x='stay_nights_bin', hue='is_canceled',
              data=data[data['hotel'] == 'City Hotel'])
plt.xlabel('Stay Nights')
plt.title('City Hotel')
plt.tight_layout()
plt.show()
```




④预定渠道

In [ ]:
```
# 预定渠道对取消率的影响
fig, ax1 = plt.subplots()
ax2 = ax1.twinx()
sns.countplot(
    x=data['distribution_channel'],
    order=data.groupby('distribution_channel')['is_canceled'].mean().index,
    ax=ax1)
ax1.set_xlabel('Distribution Channel')
ax2.plot(data.groupby('distribution_channel')['is_canceled'].mean(), 'ro-')
ax2.set_ylabel('Rate');
```



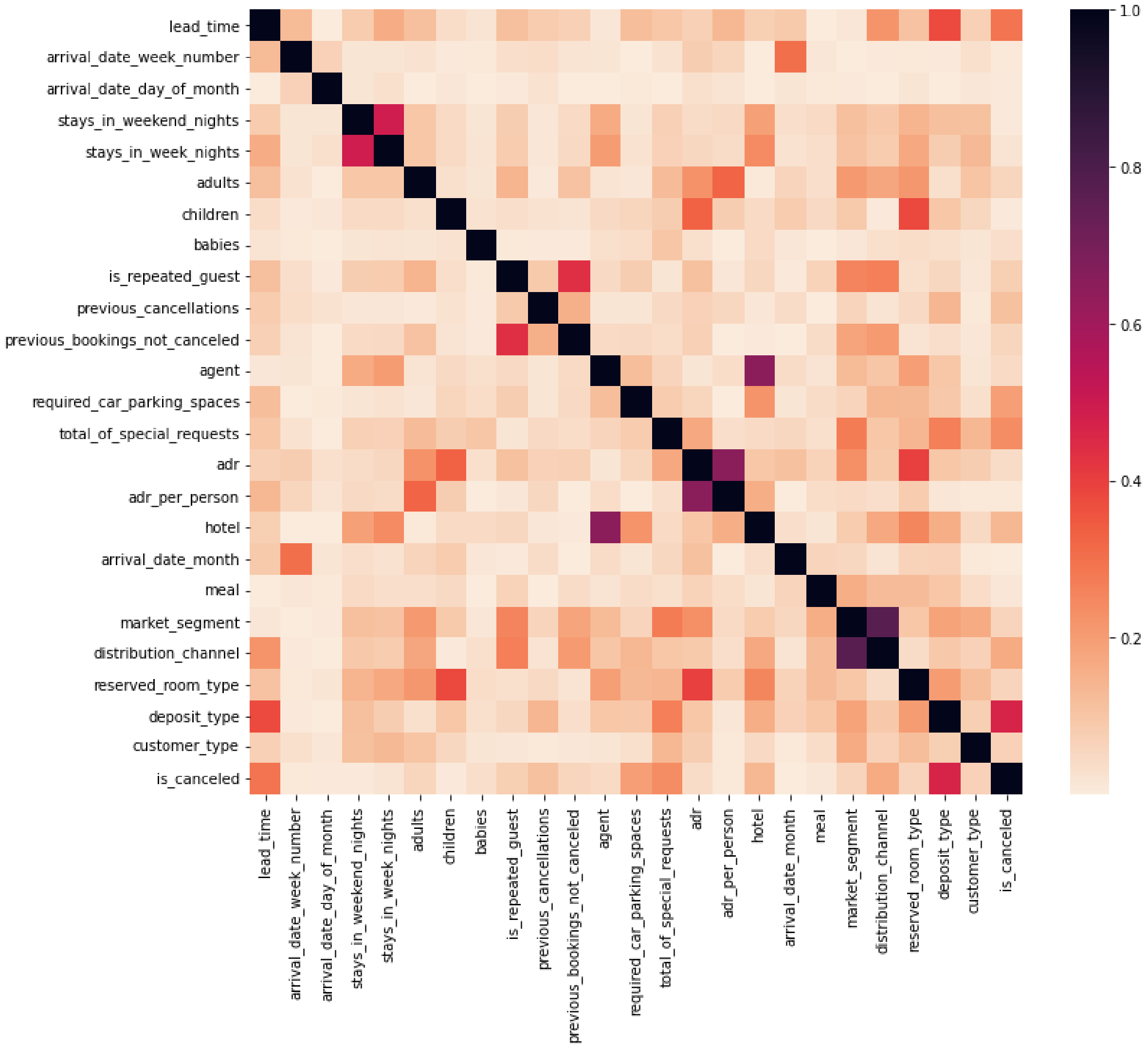

# 三、构建预测模型

In [ ]:
```
num = [
    'lead_time', 'arrival_date_week_number', 'arrival_date_day_of_month',
    'stays_in_weekend_nights', 'stays_in_week_nights', 'adults', 'children',
    'babies', 'is_repeated_guest', 'previous_cancellations',
    'previous_bookings_not_canceled', 'agent', 'required_car_parking_spaces',
    'total_of_special_requests', 'adr', 'adr_per_person'
```

```python
]

cat = [
    'hotel', 'arrival_date_month', 'meal', 'market_segment',
    'distribution_channel', 'reserved_room_type', 'deposit_type',
    'customer_type'
]

target = ['is_canceled']


ref = num+cat+target
train = data[ref]
#处理类别变量
train[cat]=train[cat].apply(LabelEncoder().fit_transform)

#处理连续变量:
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaler.fit(train[num])
train[num] = scaler.transform(train[num])
# 查看各列与取消预订的相关系数
plt.figure(figsize=(12, 10))
sns.heatmap(train.corr().abs(), cmap=sns.cm.rocket_r)
```

```
d:\anaconda\lib\site-packages\pandas\core\frame.py:3641: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  self[k1] = value[k2]
d:\anaconda\lib\site-packages\pandas\core\frame.py:3678: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  self[col] = igetitem(value, i)
```

Out[ ]:
```
<AxesSubplot:>
```




### 3.1 简单模型

In [ ]:
```
# 分离特征变量和目标变量
X = train.drop(['is_canceled'], axis=1)
y = train['is_canceled']
#划分测试集和训练集
X_train,X_test,y_train,y_test=train_test_split(X,y,stratify=y,random_state=0)

# 特征缩放
std_sca = StandardScaler()
X = std_sca.fit_transform(X)
```

In [ ]:
```
#岭回归方法
r1=RidgeClassifier(random_state=42)
r1.fit(X_train,y_train)
#print(r1.score(X_train,y_train))
print("验证集上的准确率为:{}".format(r1.score(X_test,y_test)))
```

验证集上的准确率为:0.7652992375683152

In [ ]:
```
#logistic回归方法
lr=LogisticRegression(solver='liblinear')
lr.fit(X_train,y_train)
#print(lr.score(X_train,y_train))
print("验证集上的准确率为:{}".format(lr.score(X_test,y_test)))
```

验证集上的准确率为:0.7822346670265165

In [ ]:
```
#线性SVM方法
clf = make_pipeline(StandardScaler(),LinearSVC(random_state=42))
clf.fit(X_train,y_train)
#print(clf.score(X_train,y_train))
print("验证集上的准确率为:{}".format(clf.score(X_test,y_test)));
```

```
验证集上的准确率为:0.7853046353147561
```

```
d:\anaconda\lib\site-packages\sklearn\svm\_base.py:975: ConvergenceWarning: Liblinea
r failed to converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
```

In [ ]:
```
#SGD方法
clf = make_pipeline(StandardScaler(),SGDClassifier(n_jobs=-1, random_state=42))
clf.fit(X_train,y_train)
#print(clf.score(X_train,y_train))
print("验证集上的准确率为:{}".format(clf.score(X_test,y_test)));
```

```
验证集上的准确率为:0.7663787868564874
```

In [ ]:
```
#决策树方法
clf = make_pipeline(StandardScaler(),DecisionTreeClassifier(random_state=42))
clf.fit(X_train,y_train)
#print(clf.score(X_train,y_train))
print("验证集上的准确率为:{}".format(clf.score(X_test,y_test)));
```

```
验证集上的准确率为:0.8236623709601242
```

In [ ]:
```
#Perceptron方法
r5=Perceptron(n_jobs=-1, random_state=42)
r5.fit(X_train,y_train)
#print(r5.score(X_train,y_train))
print("验证集上的准确率为:{}".format(r5.score(X_test,y_test)));
```

```
验证集上的准确率为:0.7549760475001687
```

In [ ]:
```
#K近邻方法
clf = make_pipeline(StandardScaler(),KNeighborsClassifier(n_jobs=-1))
clf.fit(X_train,y_train)
#print(clf.score(X_train,y_train))
print("验证集上的准确率为:{}".format(clf.score(X_test,y_test)));
```

```
验证集上的准确率为:0.8119560083665069
```

In [ ]:
```
#高斯NB
r8= GaussianNB()
r8.fit(X_train,y_train)
#print(r8.score(X_train,y_train))
print("验证集上的准确率为:{}".format(r8.score(X_test,y_test)));
```

```
验证集上的准确率为:0.49089130288104715
```

In [ ]:

In [ ]:
```
#随机森林方法
clf = make_pipeline(StandardScaler(),RandomForestClassifier(n_jobs=-1, random_state=
clf.fit(X_train,y_train)
#print(clf.score(X_train,y_train))
print("验证集上的准确率为:{}".format(clf.score(X_test,y_test)));
```

```
验证集上的准确率为:0.8642804129276027
```

下边用random forest 的feature_importance 来筛选特征

In [ ]:

```

```

In [ ]:

```

```

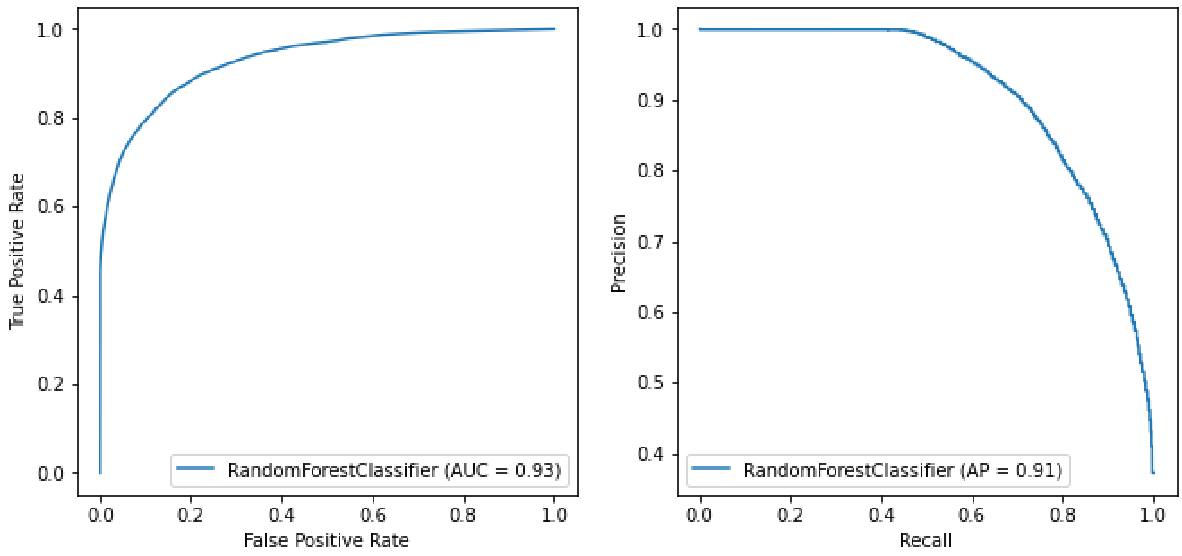
3.2 较复杂的模型

①随机森林

In [ ]:

```
#随机森林
#模型参数设置
forest = RandomForestClassifier(n_jobs=-1, random_state=42)
#模型拟合
forest.fit(X_train,y_train)
#模型预测
y_pred_rf=forest.predict(X_test)
#评价指标
print ('AUC: %.4f' % metrics.roc_auc_score(y_test,y_pred_rf))
print ('ACC: %.4f' % metrics.accuracy_score(y_test,y_pred_rf))
print ('Recall: %.4f' % metrics.recall_score(y_test,y_pred_rf))
print ('F1-score: %.4f' %metrics.f1_score(y_test,y_pred_rf))
print ('Precesion: %.4f' %metrics.precision_score(y_test,y_pred_rf))
print ('Average_Precesion: %.4f' %metrics.average_precision_score(y_test,y_pred_rf)

print('混淆矩阵为: ')
print(metrics.confusion_matrix(y_test,y_pred_rf))

rf=np.array([metrics.roc_auc_score(y_test,y_pred_rf),
            metrics.accuracy_score(y_test,y_pred_rf),
            metrics.recall_score(y_test,y_pred_rf),
            metrics.f1_score(y_test,y_pred_rf),
            metrics.precision_score(y_test,y_pred_rf),
            metrics.average_precision_score(y_test,y_pred_rf)])
print('准确率,召回率以及F1分数如下:')
print(precision_recall_fscore_support(y_test, y_pred_rf,average='binary'))
#print(forest.score(X_test,y_test))
fig, [ax_roc, ax_pr] = plt.subplots(1, 2, figsize=(11, 5))
plot_roc_curve(forest, X_test, y_test,ax=ax_roc)
plot_precision_recall_curve(forest, X_test, y_test, ax=ax_pr)
plt.show()
```

```
AUC: 0.8434
ACC: 0.8644
Recall: 0.7611
F1-score: 0.8070
Precesion: 0.8588
Average_Precesion: 0.7427
混淆矩阵为:
[[17216  1382]
 [ 2638  8406]]
准确率,召回率以及F1分数如下:
(0.8588067020841847, 0.7611372691053966, 0.8070276497695853, None)
```




In [ ]:
```
feature_list=list(train.columns.drop(['is_canceled']))
importances = list(forest.feature_importances_)
feature_importances = [(feature, round(importance, 2)) for feature, importance in z
feature_importances=pd.DataFrame(feature_importances,columns=('features','importance
feature_importances.sort_values(by=['importance'],ascending = [False],inplace=True
print(feature_importances)
```

```
                        features  importance
0                      lead_time        0.15
22                  deposit_type        0.13
15                adr_per_person        0.08
14                           adr        0.08
2      arrival_date_day_of_month        0.07
13     total_of_special_requests        0.07
1       arrival_date_week_number        0.06
9         previous_cancellations        0.05
11                         agent        0.05
4           stays_in_week_nights        0.04
19                market_segment        0.04
17            arrival_date_month        0.03
23                 customer_type        0.03
3        stays_in_weekend_nights        0.03
21            reserved_room_type        0.02
12   required_car_parking_spaces        0.02
16                         hotel        0.01
6                       children        0.01
18                          meal        0.01
5                         adults        0.01
20          distribution_channel        0.01
10 previous_bookings_not_canceled        0.00
8              is_repeated_guest        0.00
7                         babies        0.00
```

将特征重要度小于等于0.01的特征剔除，重新构建模型

In [ ]:
```
data=train.drop(columns=['hotel','children','meal','adults','distribution_channel','
```

In [ ]:
```
feature=list(data.columns.drop(['is_canceled']))
X_train,X_test,y_train,y_test=train_test_split(data[feature],data['is_canceled'],tes
```

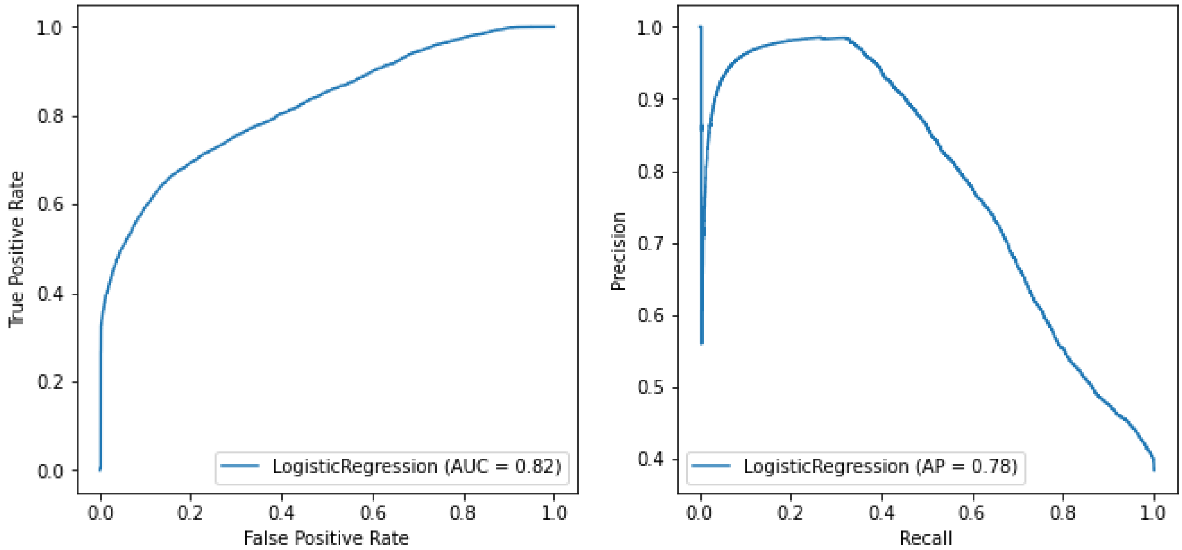
In [ ]:
```
feature
```

Out[ ]:
```
['lead_time',
 'arrival_date_week_number',
 'arrival_date_day_of_month',
 'stays_in_weekend_nights',
 'stays_in_week_nights',
 'previous_cancellations',
 'agent',
 'required_car_parking_spaces',
 'total_of_special_requests',
 'adr',
 'adr_per_person',
 'arrival_date_month',
 'market_segment',
 'reserved_room_type',
 'deposit_type',
 'customer_type']
```

②logistic 回归

In [ ]:
```
#logistic回归方法
#参数设置
model=LogisticRegression(solver='liblinear')
#模型拟合
model.fit(X_train,y_train)
#模型预测
y_pred_lr=model.predict(X_test)
#评价指标
print ('AUC: %.4f' % metrics.roc_auc_score(y_test,y_pred_lr))
print ('ACC: %.4f' % metrics.accuracy_score(y_test,y_pred_lr))
print ('Recall: %.4f' % metrics.recall_score(y_test,y_pred_lr))
print ('F1-score: %.4f' %metrics.f1_score(y_test,y_pred_lr))
print ('Precesion: %.4f' %metrics.precision_score(y_test,y_pred_lr))
print ('Average_Precesion: %.4f' %metrics.average_precision_score(y_test,y_pred_lr)

print('混淆矩阵为：')
print(metrics.confusion_matrix(y_test,y_pred_lr))

lr=np.array([metrics.roc_auc_score(y_test,y_pred_lr),
            metrics.accuracy_score(y_test,y_pred_lr),
            metrics.recall_score(y_test,y_pred_lr),
            metrics.f1_score(y_test,y_pred_lr),
            metrics.precision_score(y_test,y_pred_lr),
            metrics.average_precision_score(y_test,y_pred_lr)])
print('准确率,召回率以及F1度量如下:')
print(precision_recall_fscore_support(y_test, y_pred_lr,average='binary'))
#print('测试集得分：',lr.score(X_test,y_test))
fig, [ax_roc, ax_pr] = plt.subplots(1, 2, figsize=(11, 5))
plot_roc_curve(model, X_test, y_test,ax=ax_roc)
plot_precision_recall_curve(model, X_test, y_test, ax=ax_pr)
plt.show()
```

```
AUC: 0.7213
ACC: 0.7796
Recall: 0.4847
F1-score: 0.6237
Precesion: 0.8747
Average_Precesion: 0.6181
混淆矩阵为：
[[21234   931]
 [ 6908  6497]]
准确率,召回率以及F1度量如下：
(0.874663435648896, 0.4846698992913092, 0.6237219795516729, None)
```



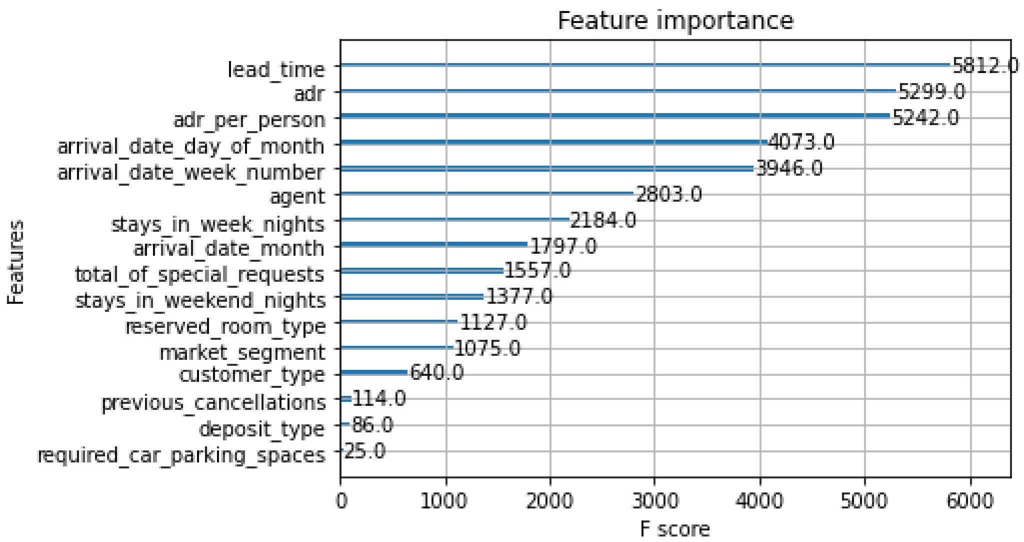

### ③xgboost 模型

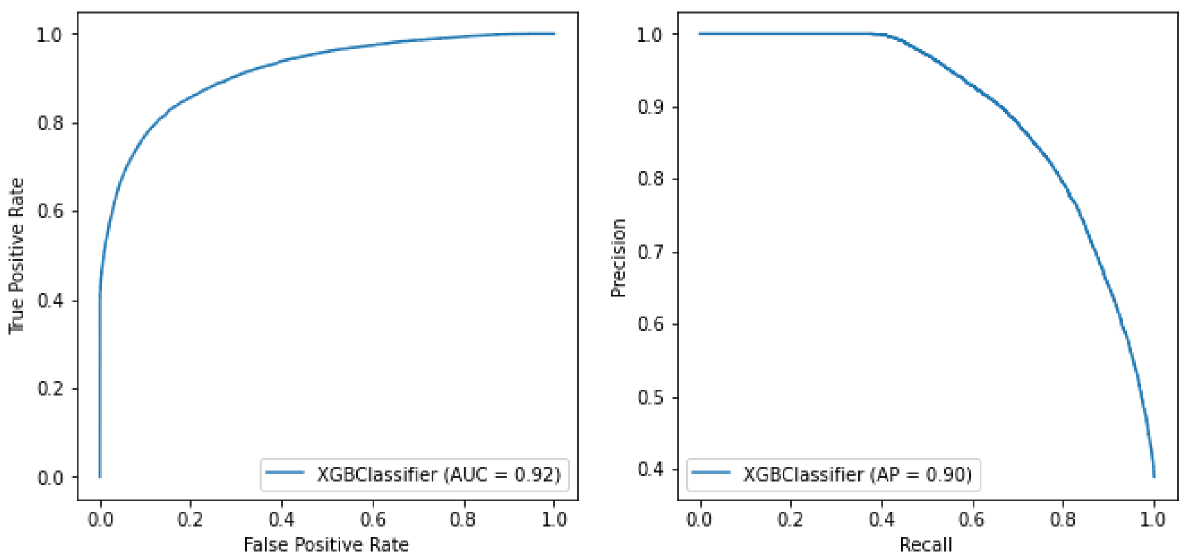
In [ ]:

```
#xgboost的参考资料见https://www.cnblogs.com/harekizgel/p/7683803.html和https://blog.

import xgboost as xgb
# 初始化模型
model = xgb.XGBClassifier(n_estimators=200,max_depth=9,learning_rate=0.3,booster='g
#scale_pos_weight是样本集为不平衡数据集时给样本设置权重的参数

# 拟合模型
model.fit(X_train, y_train)

# 使用模型预测
y_pred_xgb = model.predict(X_test)

# 评价标准
from sklearn import metrics
print ('AUC: %.4f' % metrics.roc_auc_score(y_test,y_pred_xgb))
print ('ACC: %.4f' % metrics.accuracy_score(y_test,y_pred_xgb))
print ('Recall: %.4f' % metrics.recall_score(y_test,y_pred_xgb))
print ('F1-score: %.4f' %metrics.f1_score(y_test,y_pred_xgb))
print ('Precesion: %.4f' %metrics.precision_score(y_test,y_pred_xgb))
print ('Average_Precesion: %.4f' %metrics.average_precision_score(y_test,y_pred_xgb

print('混淆矩阵为：')
print(metrics.confusion_matrix(y_test,y_pred_xgb))

xgb=np.array([metrics.roc_auc_score(y_test,y_pred_xgb),
            metrics.accuracy_score(y_test,y_pred_xgb),
            metrics.recall_score(y_test,y_pred_xgb),
            metrics.f1_score(y_test,y_pred_xgb),
            metrics.precision_score(y_test,y_pred_xgb),
```

```
            metrics.average_precision_score(y_test,y_pred_xgb)])
#特征重要性排序
from xgboost import plot_importance as plot_importance_xgb
plot_importance_xgb(model)
plt.show()
fig, [ax_roc, ax_pr] = plt.subplots(1, 2, figsize=(11, 5))
plot_roc_curve(model, X_test, y_test,ax=ax_roc)
plot_precision_recall_curve(model, X_test, y_test, ax=ax_pr)
plt.show()
```

```
AUC: 0.8083
ACC: 0.7870
Recall: 0.8945
F1-score: 0.7600
Precesion: 0.6606
Average_Precesion: 0.6307
混淆矩阵为:
[[16004  6161]
 [ 1414 11991]]
```



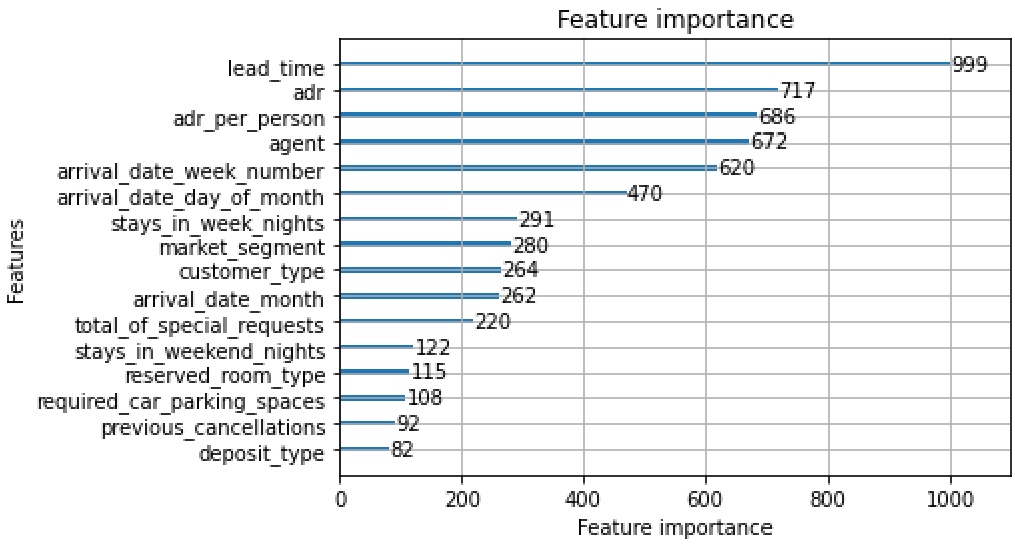



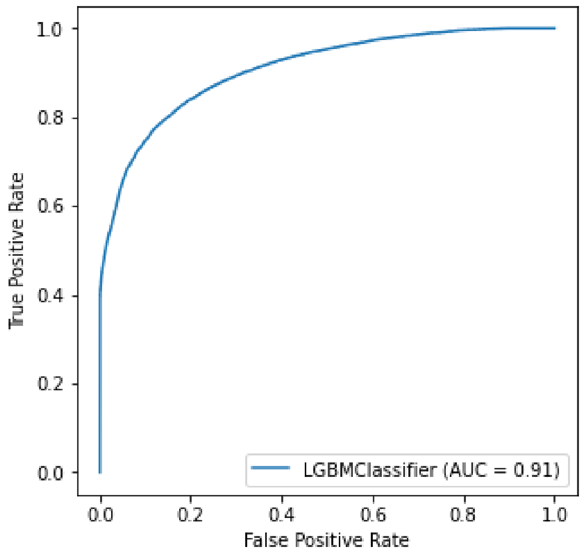

④lightgbm 模型

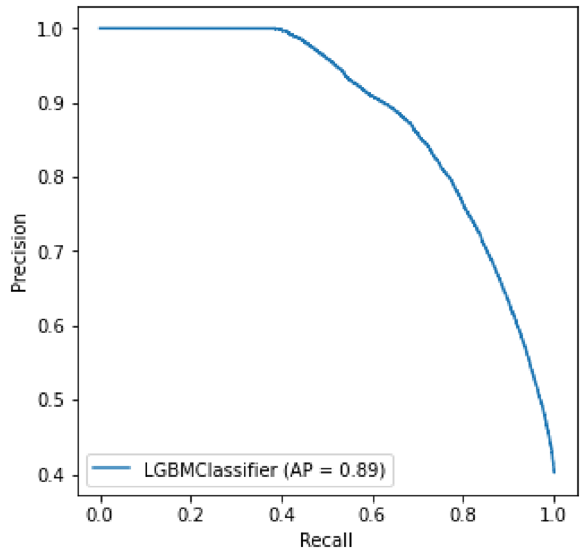
```
In [ ]: #lightgbm的参考资料见https://blog.csdn.net/zhong_ddbb/article/details/107285482?ops_
#和https://www.cnblogs.com/bjwu/p/9307344.html
from lightgbm import LGBMClassifier
model = LGBMClassifier(
    max_depth=9,
    learning_rate=0.1,
```

```python
    n_estimators=200, # 使用多少个弱分类器
    objective='binary',
    booster='gbtree',
    min_child_weight=2,
    subsample=0.8,
    colsample_bytree=0.8,
    reg_alpha=0,
    reg_lambda=1,
    seed=0 # 随机数种子
)
model.fit(X_train,y_train)

# 对测试集进行预测
y_pred_lgb = model.predict(X_test)
# 评价标准
print ('AUC: %.4f' % metrics.roc_auc_score(y_test,y_pred_lgb))
print ('ACC: %.4f' % metrics.accuracy_score(y_test,y_pred_lgb))
print ('Recall: %.4f' % metrics.recall_score(y_test,y_pred_lgb))
print ('F1-score: %.4f' %metrics.f1_score(y_test,y_pred_lgb))
print ('Precesion: %.4f' %metrics.precision_score(y_test,y_pred_lgb))
print ('Average_Precesion: %.4f' %metrics.average_precision_score(y_test,y_pred_lgb

print('混淆矩阵为：')
print(metrics.confusion_matrix(y_test,y_pred_lgb))

lgb=np.array([metrics.roc_auc_score(y_test,y_pred_lgb),
            metrics.accuracy_score(y_test,y_pred_lgb),
            metrics.recall_score(y_test,y_pred_lgb),
            metrics.f1_score(y_test,y_pred_lgb),
            metrics.precision_score(y_test,y_pred_lgb),
            metrics.average_precision_score(y_test,y_pred_lgb)])
#特征重要性排序
from lightgbm import plot_importance as plot_importance_lgb
plot_importance_lgb(model)
plt.show()
fig, [ax_roc, ax_pr] = plt.subplots(1, 2, figsize=(11, 5))
plot_roc_curve(model, X_test, y_test,ax=ax_roc)
plot_precision_recall_curve(model, X_test, y_test, ax=ax_pr)
plt.show()
```

```
d:\anaconda\lib\site-packages\dask\dataframe\utils.py:14: FutureWarning: pandas.uti
l.testing is deprecated. Use the functions in the public API at pandas.testing inste
ad.
  import pandas.util.testing as tm
[LightGBM] [Warning] Unknown parameter: booster
AUC: 0.8135
ACC: 0.8427
Recall: 0.6949
F1-score: 0.7691
Precesion: 0.8610
Average_Precesion: 0.7133
混淆矩阵为：
[[20661  1504]
 [ 4090  9315]]
```



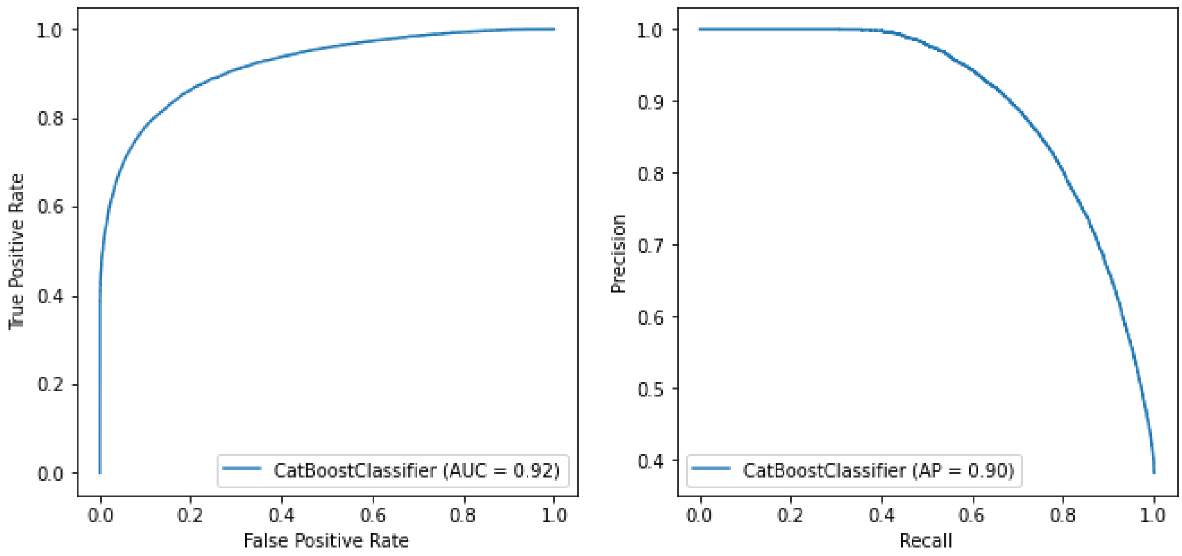

⑤catboost 模型

In [ ]:

```
#catboost的参考资料见https://www.biaodianfu.com/catboost.html#CatBoost%E4%BD%BF%E7%9

import catboost as cb


#模型参数配置
model = cb.CatBoostClassifier(iterations=1000, depth=10, learning_rate=0.3, loss_fu
#logging_level='Silent'控制输出日志信息

#模型拟合
model.fit(X_train, y_train,)#cat_features=[0,2,5]用来标记分类特征

# 使用模型预测
y_pred_cab = model.predict(X_test)#预测类别
#y_pred_probs = model.predict_proba(X_test)#预测类别的概率

# 评价标准
print ('AUC: %.4f' % metrics.roc_auc_score(y_test,y_pred_cab))
print ('ACC: %.4f' % metrics.accuracy_score(y_test,y_pred_cab))
print ('Recall: %.4f' % metrics.recall_score(y_test,y_pred_cab))
print ('F1-score: %.4f' %metrics.f1_score(y_test,y_pred_cab))
print ('Precesion: %.4f' %metrics.precision_score(y_test,y_pred_cab))
print ('Average_Precesion: %.4f' %metrics.average_precision_score(y_test,y_pred_cab
```

```
print('混淆矩阵为：')
print(metrics.confusion_matrix(y_test,y_pred_cab))

cab=np.array([metrics.roc_auc_score(y_test,y_pred_cab),
            metrics.accuracy_score(y_test,y_pred_cab),
            metrics.recall_score(y_test,y_pred_cab),
            metrics.f1_score(y_test,y_pred_cab),
            metrics.precision_score(y_test,y_pred_cab),
            metrics.average_precision_score(y_test,y_pred_cab)])
#特征重要性排序
feature_list=list(data.columns.drop(['is_canceled']))
importances_list = list(model.feature_importances_)
feature_importances = [(feature, round(importance, 3)) for feature, importance in z
feature_importances=pd.DataFrame(feature_importances,columns=('features','importance
feature_importances.sort_values(by=['importance'],ascending = [False],inplace=True
print(feature_importances)

fig, [ax_roc, ax_pr] = plt.subplots(1, 2, figsize=(11, 5))
plot_roc_curve(model, X_test, y_test,ax=ax_roc)
plot_precision_recall_curve(model, X_test, y_test, ax=ax_pr)
plt.show()
```

```
AUC: 0.8383
ACC: 0.8558
Recall: 0.7674
F1-score: 0.8004
Precesion: 0.8364
Average_Precesion: 0.7295
混淆矩阵为：
[[20153  2012]
 [ 3118 10287]]
                        features  importance
0                      lead_time      12.360
14                  deposit_type       9.154
2      arrival_date_day_of_month       8.832
9                            adr       7.622
1        arrival_date_week_number      7.468
10                adr_per_person       7.420
6                          agent       6.714
4           stays_in_week_nights       6.292
8      total_of_special_requests       6.096
11            arrival_date_month       5.272
3        stays_in_weekend_nights       4.919
12                market_segment       4.497
7    required_car_parking_spaces       4.288
15                 customer_type       3.285
5         previous_cancellations       3.037
13            reserved_room_type       2.742
```




## 综合比较

```
In [ ]: df = pd.DataFrame(np.vstack((lr, rf, xgb,lgb,cab)),
                          index=['logisticRegression','RandomForest','xgboost','lightgbm','ca
                          columns=['AUC','Accuracy','Recall','F1-score','Precesion','Average_
        print(df)
```

```
                         AUC  Accuracy    Recall  F1-score  Precesion  \
logisticRegression  0.721333  0.779618  0.484670  0.623722   0.874663
RandomForest        0.843414  0.864382  0.761137  0.807028   0.858807
xgboost             0.808278  0.787040  0.894517  0.759958   0.660588
lightgbm            0.813518  0.842733  0.694890  0.769072   0.860985
catboost            0.838313  0.855777  0.767400  0.800420   0.836409

                    Average_Precesion
logisticRegression           0.618132
RandomForest                 0.742665
xgboost                      0.630660
lightgbm                     0.713275
catboost                     0.729519
```