湖北汽车工业学院

《大数据综合应用实训》报告

课题名称：　大数据综合应用实训

专　　业：　计算机科学与技术

班　　级：

学　　号：

姓　　名：

校内教师：　田俐

成　　绩：

2023 年　4 月 20 日—— 2023 年　5 月 31 日

实训报告书写要求

1、严禁抄袭。如有抄袭（30%以上，抄袭者与被抄袭者）一律为不及格。

2、篇幅不少于 20 页（其中代码内容不超 10 页，总结不少于 1 页）。

3、请按照实训报告内容框架进行书写，字体采用宋体、小四号字，正文采用 1.5 倍行距，要求排版整齐。

4、报告文件名为：学号_姓名.docx，例如：20070230103_张三.docx，统一提交至学习通作业中。

5、实训内容：系统实现部分结合文字描述放核心代码即可，每段代码前都应有文字描述，不能都是代码。

一、 实训项目概述

关于酒店预订数据集的探索

【EDA+五折交叉验证决策树】

本实训报告介绍了关于酒店预订数据集的探索分析和决策树模型预测的过程和结果。本报告旨在通过对酒店预订数据集的探索和决策树模型的应用，提高数据分析和机器学习的能力和技巧。

本项目包含：

1.数据处理

对数据集进行缺失值处理、异常值检测、特征编码等操作，使数据集符合后续分析和建模的要求。

2.数据探索性分析

利用可视化工具和统计方法，对数据集进行描述性分析，探索数据的分布特征、相关性、异常情况等。

3.网格搜索对决策树进行模型参数调优

使用网格搜索方法，对决策树模型的参数进行优化，寻找最佳的参数组合，提高模型的预测性能。

4.基于五折交叉验证的决策树模型预测

使用五折交叉验证方法，对决策树模型进行评估和测试，计算模型的准确率、召回率、F1 值、ROC 曲线等指标，分析模型的优缺点。

数据与背景描述

背景描述

在线酒店预订是一种为世界各地的旅行者预订住宿的便捷且流行的方式。

它允许客户比较各种酒店、住宅或度假屋的价格、设施、位置和评论，然后单击几下即可在线预订。 在线酒店预订平台针对任何预算和偏好提供广泛的选择，以及航班、汽车租赁、出租车和景点等附加服务。 在线酒店预订还为客户提供了灵活性和安全性，因为在大多数情况下他们可以取消或修改预订而不会受到处罚，并受益于客户支持和忠诚度计划。在线酒店预订渠道已经极大地改变了预订的可能性和客户的行为。

酒店预订取消的典型原因包括计划的改变、日程安排的冲突等。对酒店客人来说，可以选择免费或低价取消预订，因此客人们更容易取消预订。但对酒店来说，这是一个不利于酒店收入和利润的因素，这是一个需要解决的问题。酒店预订取消的影响有：

- 对酒店的收入和利润造成损失，降低酒店的经营效率和竞争力。
- 对酒店的信誉和口碑造成负面影响，降低客户的满意度和忠诚度。
- 对酒店的资源和人力造成浪费，增加酒店的运营成本和管理难度。

酒店预订取消的解决办法有：

- 设定合理的取消政策和收费标准，提高客户的预订意愿和责任感。
- 加强与客户的沟通和服务，及时了解客户的需求和变化，提供灵活的预订修改和调整方案。
- 建立客户关系管理系统，维护老客户的忠诚度，吸引新客户的信任度，提升酒店的品牌形象。
- 利用数据分析和机器学习，预测客户的取消概率和行为，制定相应的营销策略和优惠措施。

为了实现利用数据分析和机器学习，预测客户的取消概率和行为，制定相应

的营销策略和优惠措施，我们需要有一些可靠的数据来源。幸运的是，我们有一个关于酒店预订的数据集，它包含了 36275 条预订记录，每条记录有 19 个特征，涵盖了客户的基本信息、预订情况、入住时间、房间类型、价格等方面。这个数据集可以帮助我们分析客户的预订行为和取消原因，从而提出合理的建议和解决方案。

下面，我们将对这个数据集进行详细的说明。

数据说明

column	*列名*
Booking_ID	每个预订的唯一标识符
no_of_adults	成人的数量
no_of_children	儿童的数量
no_of_weekend_nights	客人入住或预订入住酒店的周末晚数（周六或周日）
no_of_week_nights	客人在酒店住宿或预订住宿的周晚数（周一至周五）
type_of_meal_plan	客户预订的膳食计划的类型
required_car_parking_space	顾客是否需要一个停车位？(0–不，1–是)
room_type_reserved	顾客预订的房间类型。这些值是由 INN 酒店集团加密（编码）的
lead_time	预订日期和抵达日期之间的天数
arrival_year	抵达日期的年份
arrival_month	抵达日期的月份

arrival_date	该月的日期
market_segment_type	市场部分的指定
repeated_guest	该客户是否为重复客人？(0 – 否, 1– 是)
no_of_previous_cancellations	在当前预订之前，客户取消的先前预订的数量
no_of_previous_bookings_not_canceled	在当前预订前未被客户取消的先前预订的数量
avg_price_per_room	每天预订的平均价格；房间的价格是动态的。(单位：欧元)
no_of_special_requests	客户提出的特殊要求的总数（例如，高楼层，从房间看风景等）
booking_status	表示预订是否被取消的标志

二、 数据预处理

【注】*为突出重点，部分代码或输出部分已省略，详细内容见源代码附录。*

数据预处理是数据分析或数据挖掘之前的一个重要且必要的步骤。 它旨在提高数据质量，使其适用于用于分析或挖掘的软件或方法。

这部分使用 LabelEncoder 对分类特征进行编码，并使用 train_test_split 将数据拆分为训练集和测试集。

导入并检查数据

这一步是将文件中的数据加载到一个可以操作和分析的数据结构中，比如 pandas dataframe。 这部分使用 pandas 导入数据并检查缺失值、重复项和基

本统计信息，还导入了 matplotlib 和 seaborn 用于数据可视化。它还涉及检查数据的基本信息，例如行数和列数、数据类型、列名等。

导入数据

```
import pandas as pd    # 导入 pandas 库，用于数据处理
df = pd.read_csv('./Hotel Reservations.csv') # 读取酒店预订数据集
```

检查数据

数据无缺失，无重复

```
df.info()              # 显示数据集的基本信息，如列名、数据类型、缺失值等
```

查看缺失值

此步骤是识别和处理数据中的任何缺失值，例如用平均值或中值替换它们，删除缺失值过多的行或列，或插补 他们用一些算法。

```
df.duplicated().sum()              # 统计数据集中的重复行数
```

查看统计信息

这一步是计算并显示数据的一些描述性统计信息，如均值、标准差、最小值、最大值、四分位数等，有助于理解 数据的分布和特征。

```
df.describe().T              # 显示数据集中数值型变量的描述性统计，如均值、标准差、最大值、最小值等
```

三、 数据分析与可视化

这一步是探索数据中不同变量之间的关系，例如使用相关分析、假设检验或聚类方法。 它还涉及创建各种图表和图形以直观地显示数据中的模型和趋势。

这部分使用各种图表来探索数据并找到可能的可视化模型，还使用了一些描述性统计和相关性分析。

EDA　数据含义与分析目的

EDA，即数据探索性分析，是通过了解数据集，了解变量间的相互关系以及变量与预测值之间的关系，从而帮助我们后期更好地进行特征工程和建立模型，是数据挖掘中十分重要的一步。

数据含义

首先看看数据都有哪些，数据一共 19 列：

预定 ID	顾客数量	顾客预定天数	顾客需求类数据	日期、时间类型数据	预定方法	是否为历史用户	本次前客户是否取消数目	预定房价的平均价格	是否被取消
是唯一标识符，仅用于区分数据。	包括成人数量和儿童数量两列。	分为工作日和周末两列。	用餐类型，停车位，房间类型，顾客特殊要求数量四列。	预定与抵达日间隔天数，抵达日期年份、月份，抵达日期四列。	在线、离线。	是/否。	分为取消数目、未取消数目两列。	价格。	是/否。

明确目的

然后明确数据探索性分析的目的：我们想找出是否取消预定与上述其他特征是否存在一定的关系。所以我们可以进行对比分析，这里只进行简单的分析，变量间关系暂不分析。

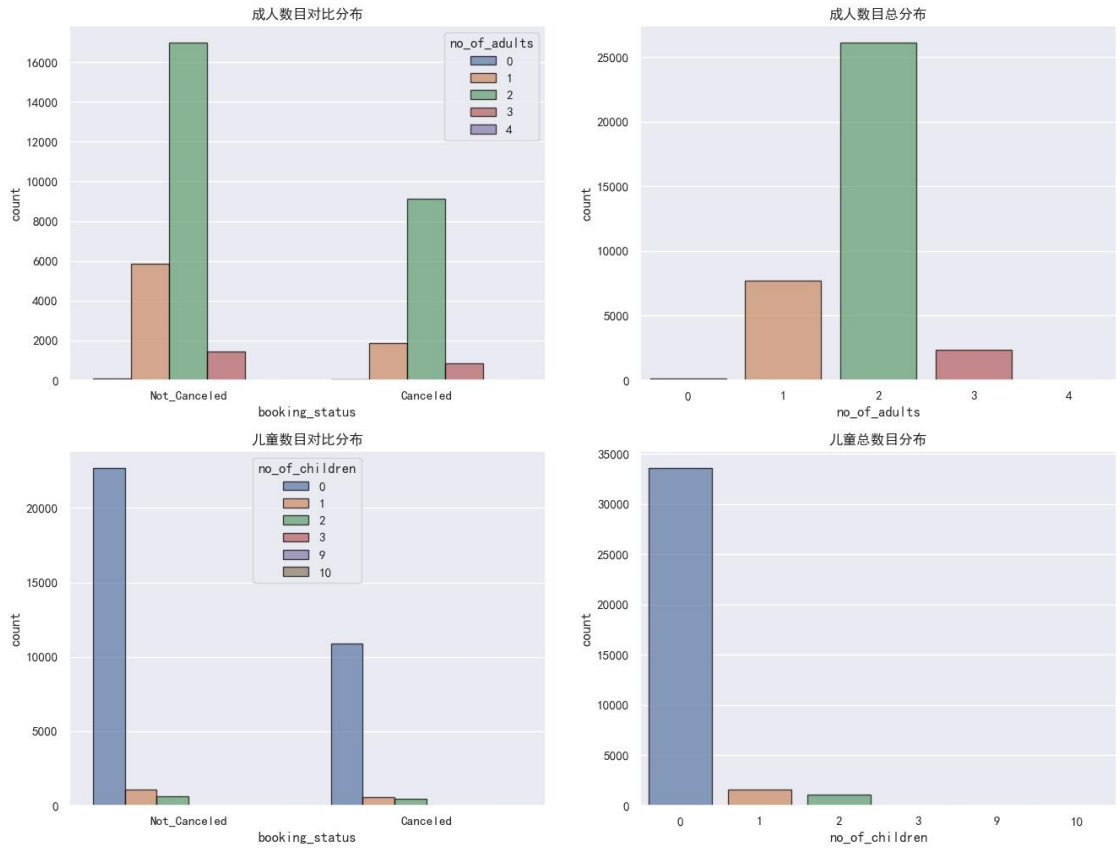
成人、儿童数目的分析

df.no_of_children.value_counts()

```
import matplotlib.pyplot as plt                    # 导入 matplotlib 库，用于绘制图形
import numpy as np                                 # 导入 numpy 库，用于数值计算
import seaborn as sns                              # 导入 seaborn 库，用于绘制更美观的图形
sns.set()                                          # 设置 seaborn 为默认风格
plt.rcParams['font.sans-serif'] = ['SimHei']   #设置中文字体为黑体
plt.rcParams['axes.unicode_minus'] = False #正常显示负号
# %matplotlib inline                               # 在 Jupyter Notebook 中显示图形


# 绘制成人和儿童数量与预订状态的关系图

plt.figure(figsize = (16, 12))                     # 设置图形大小为 16 x 12 英寸
plt.suptitle("成人、儿童顾客的数目",fontweight="bold", fontsize=30)
# 设置总标题为“成人、儿童顾客的数目”，加粗字体，字号为 30

plt.subplot(2,2,1)                                 # 在 2 x 2 的子图网格中绘制第一个子图
plt.gca().set_title('成人数目对比分布')
sns.countplot(x = 'booking_status', hue = 'no_of_adults',    edgecolor="black",
alpha=0.7, data = df)
# 使用 seaborn 的 countplot 函数绘制成人数量在不同预订状态下的柱状图，设置边框
颜色为黑色，透明度为 0.7


plt.subplot(2,2,2)
plt.gca().set_title('成人数目总分布')
sns.countplot(x = 'no_of_adults', edgecolor="black", alpha=0.7,data = df)
```

```
plt.subplot(2,2,3)
plt.gca().set_title('儿童数目对比分布')
sns.countplot(x = 'booking_status', hue = 'no_of_children',    edgecolor="black",
alpha=0.7, data = df)

plt.subplot(2,2,4)
plt.gca().set_title('儿童总数目分布')
sns.countplot(x = 'no_of_children',      edgecolor="black", alpha=0.7,data = df)
plt.show()
```

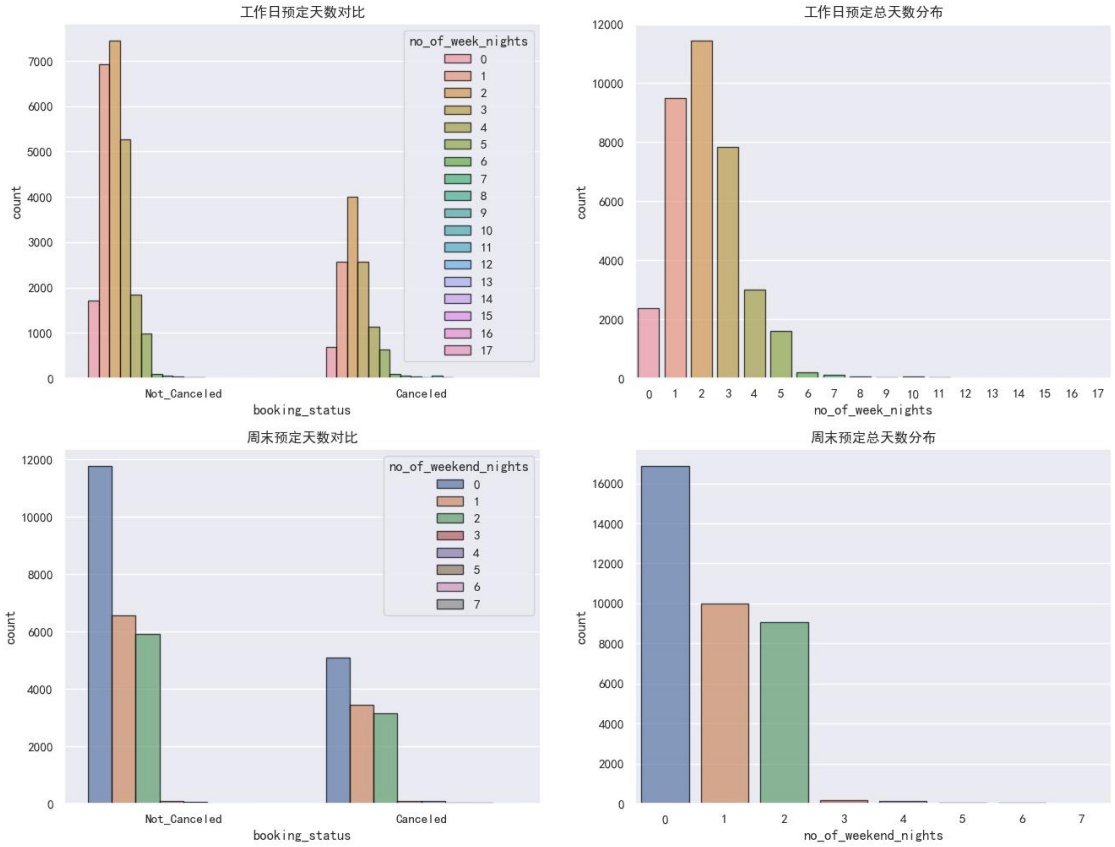
成人、儿童顾客的数目



顾客预定天数的分布

```
# 绘制周末晚上和工作日晚上数量与预订状态的关系图
```

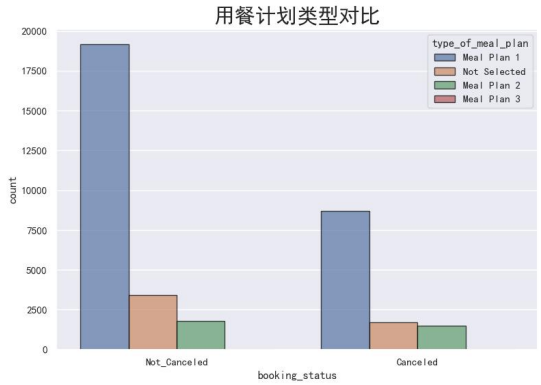
顾客预定天数的分布

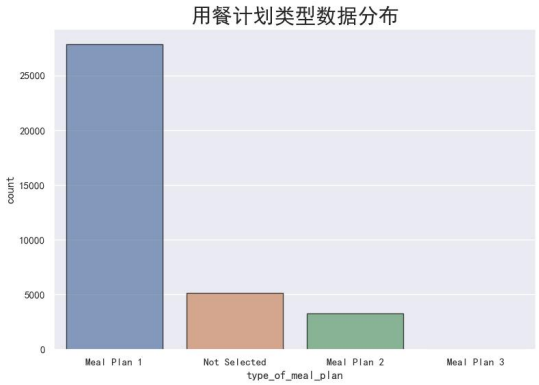


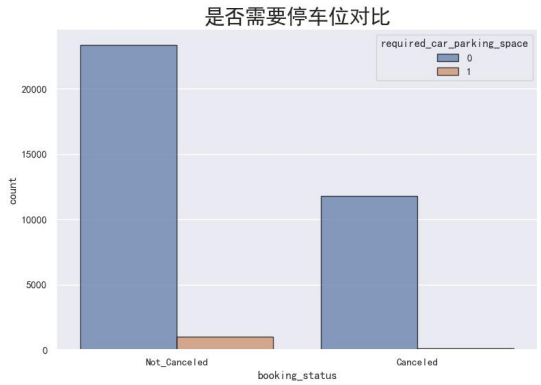
顾客需求类数据分析

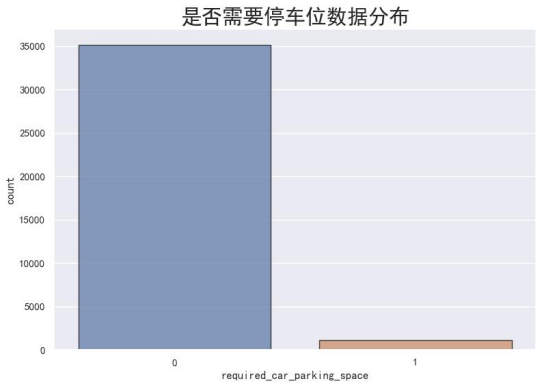
```
# 绘制客户需求类数据与预订状态的关系图
```

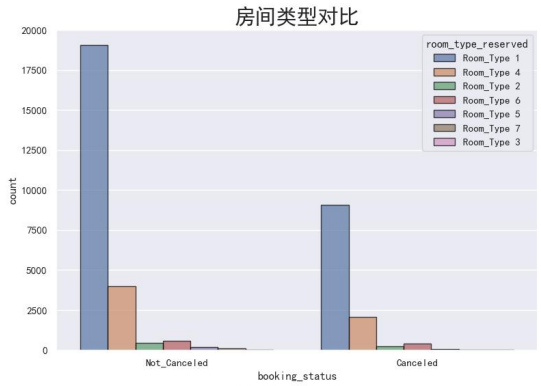
顾客需求类数据分析

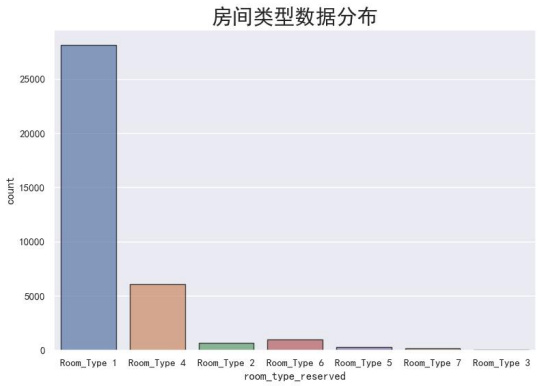


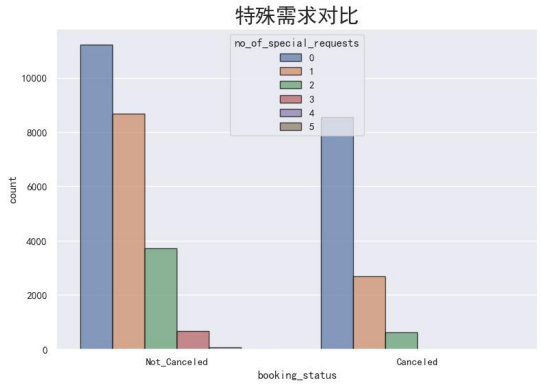


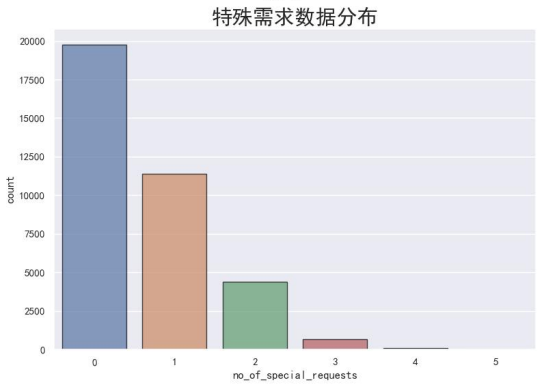












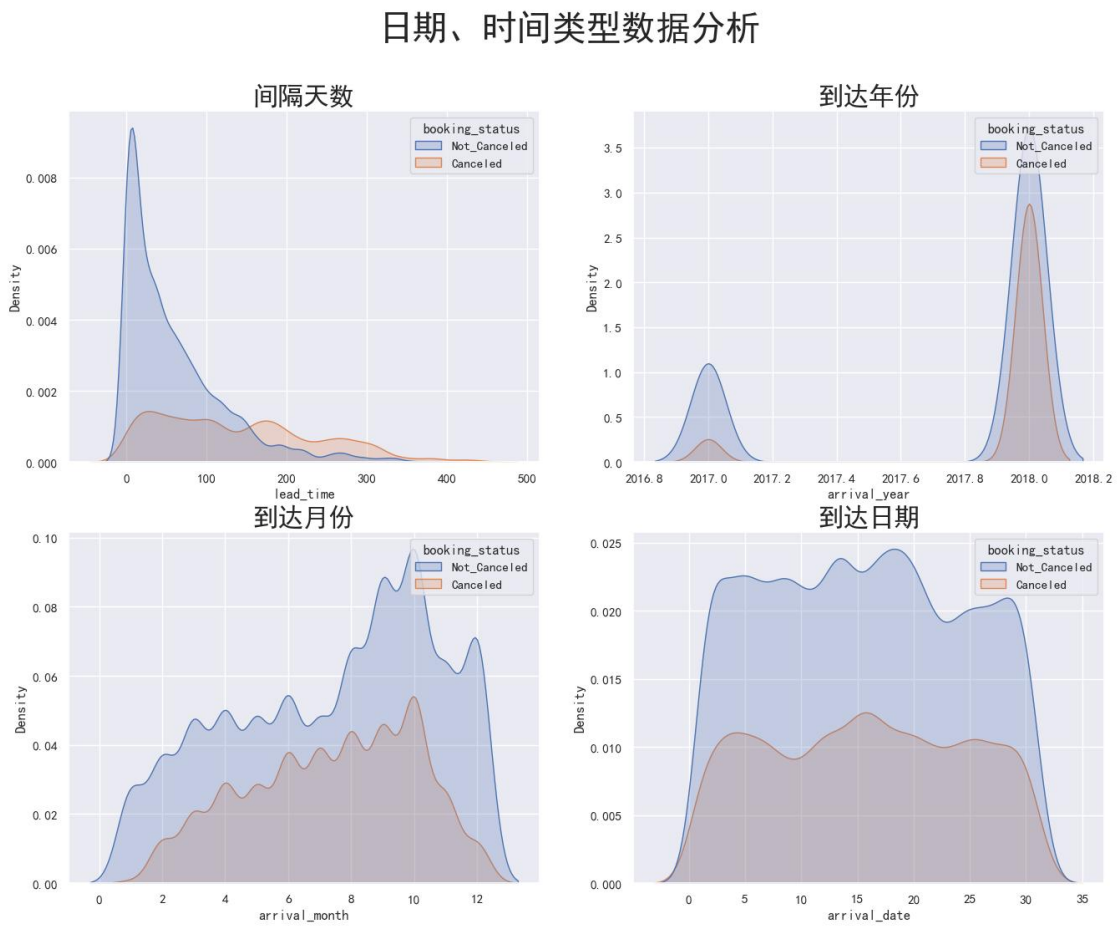
由图一可发现,用餐计划 3 基本上没有人选择 应对措施：对膳食计划进行修订,

如调整膳食计划 3。

日期、时间类型数据分析

日期、时间类型数据：预定与抵达日间隔天数 lead_time，抵达日期年份 arrival_year、月份 arrival_month，抵达日期四列 arrival_date。

```
# 绘制周末晚上和工作日晚上数量与预订状态的关系图
```



这里图 1 的间隔天数可看出预定时间越早就更容易取消。

预定住宿日期为 1 月，12 月的时候，取消概率最小；其他月基本在一定范围内波动；且自然日基本也在一定范围内波动。

对应措施 ： 建议在 2-11 月时，可适当降低客房单价，以稳定客户预定；1 月和 12 月，可根据实际情况提高单价；

房单价越高，取消预定概率越大。

其他数据分析

预定方法(在线、离线等)	是否为历史用户	取消数目	未取消数目	预定房价的平均价格
market_segment_type	repeated_guest	no_of_previous_cancellations	no_of_previous_bookings_not_canceled	avg_price_per_room

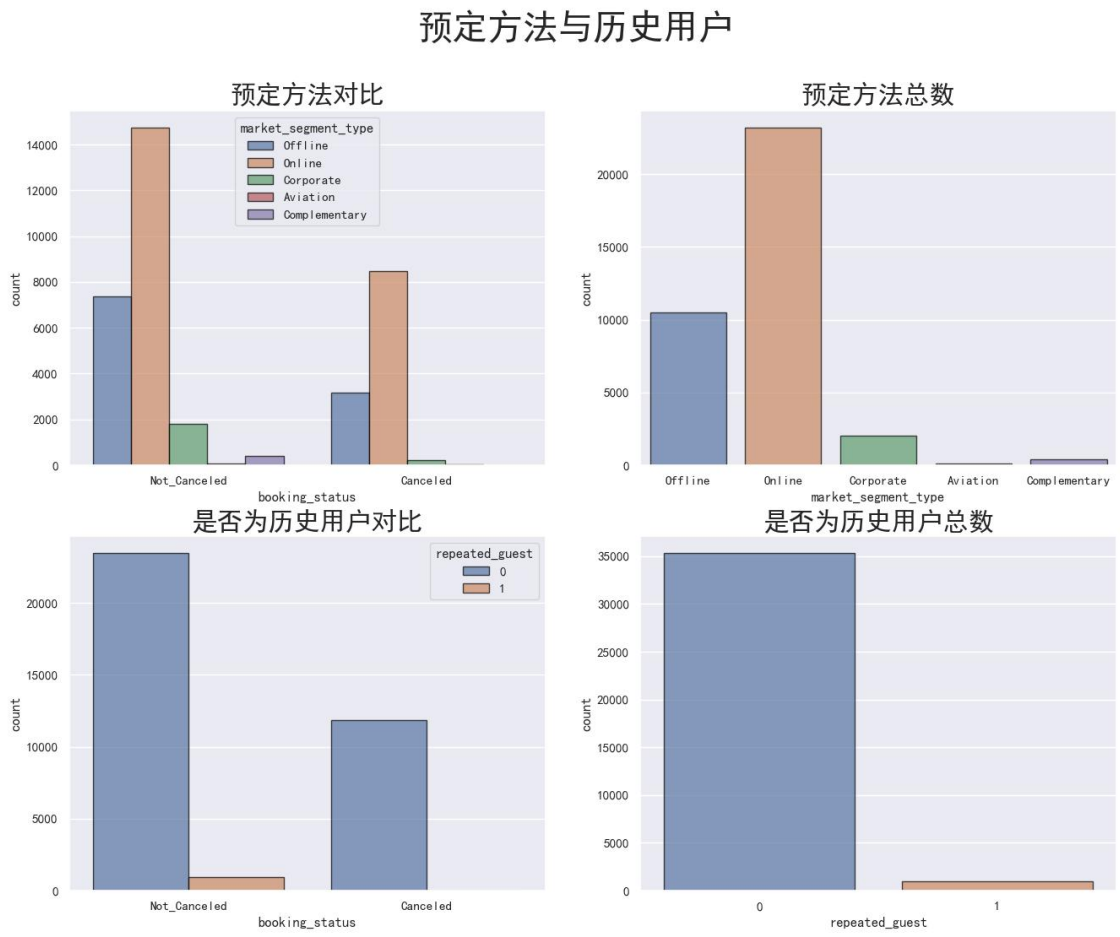
#offline 线下

#online 线上

#corporate 企业联合

#aviation 飞机票联合预定

#complementary



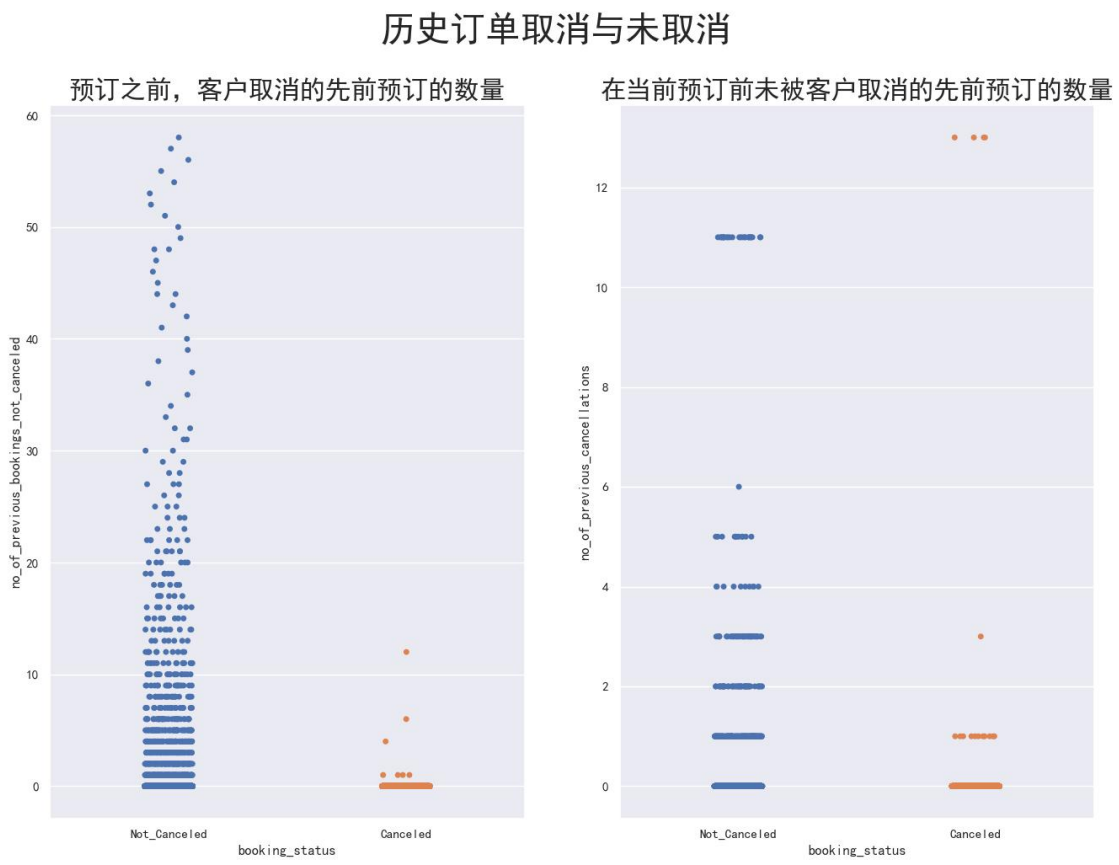
上图历史用户 repeated_guest 0 是新用户，1 是老用户。

是否为历史用户对比图说明老用户基本上不取消。

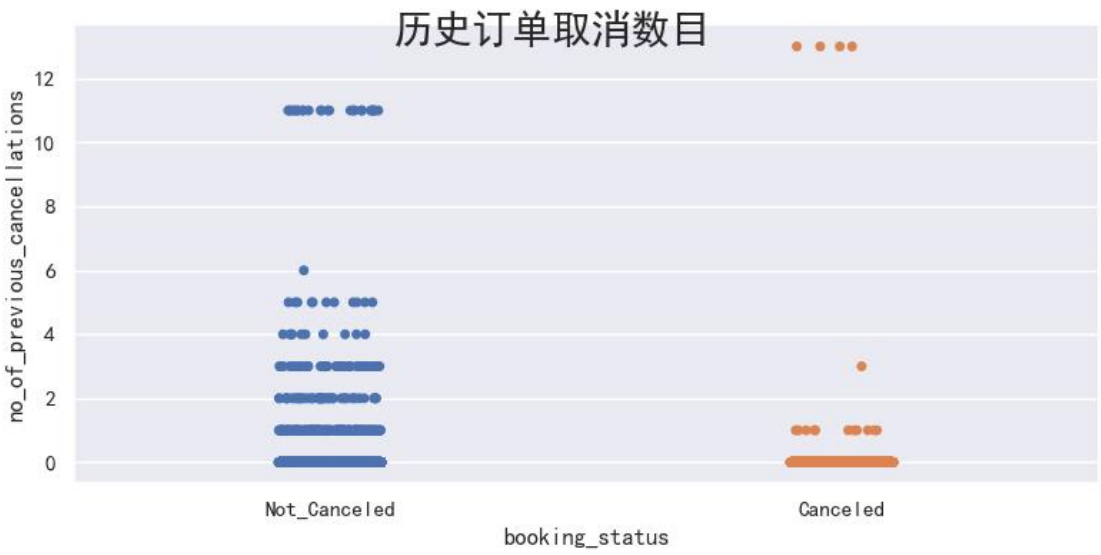
相对应措施：老用户取消预定概率最小，要维护好老用户的入住体验。

公司合作预定的取消概率最小。

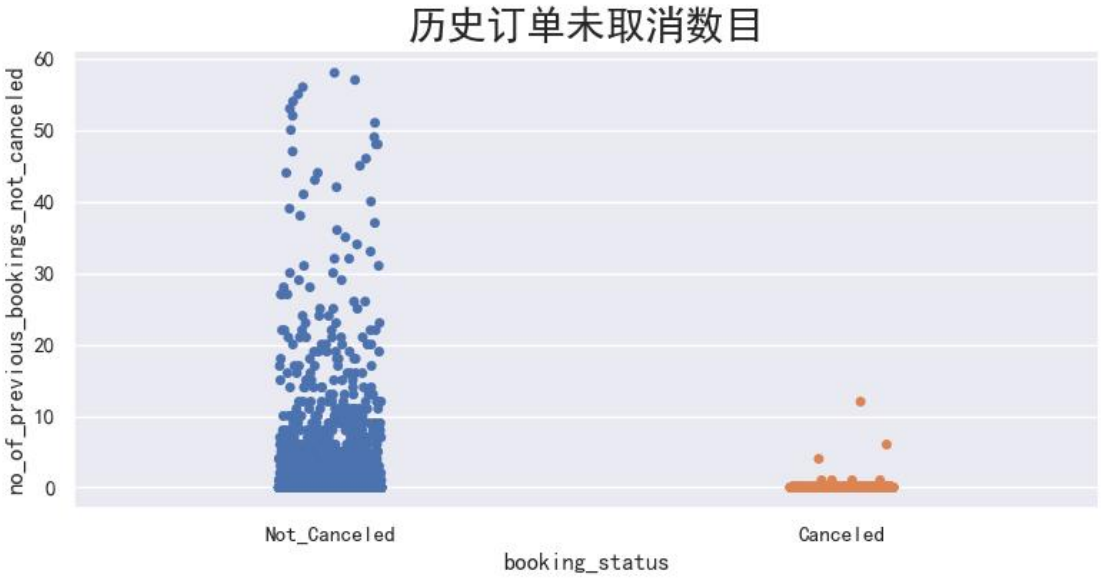
对应措施 ： 做好公司团购预定合作，该类服务更不易取消。

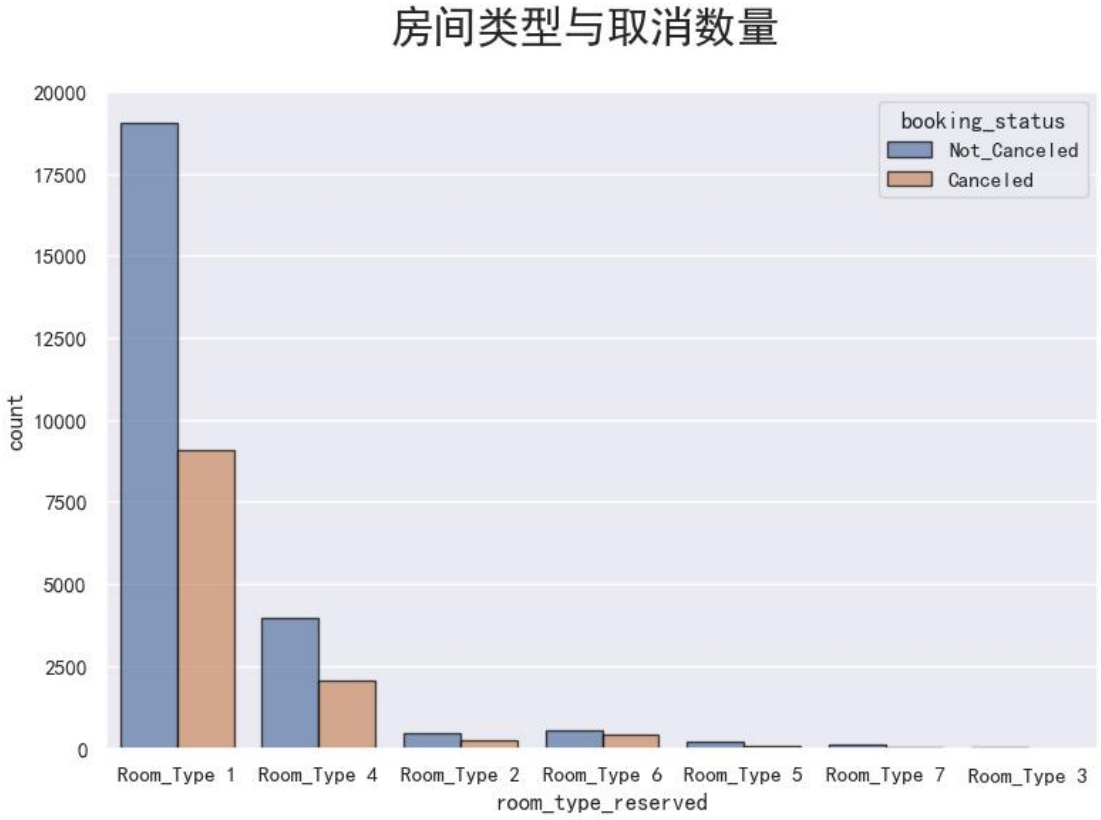



```
ax = sns.catplot('booking_status', 'no_of_previous_cancellations',height=4, aspect=2,
data=df)
ax.fig.suptitle("历史订单取消数目",
                fontsize=20, fontdict={"weight": "bold"})
plt.show()
```




```
ax2 = sns.catplot('booking_status', 'no_of_previous_bookings_not_canceled'
                  ,height=4, aspect=2
                  ,
                  data=df)
plt.title("历史订单未取消数目",
                  fontsize=20,
                 )
plt.show()
```

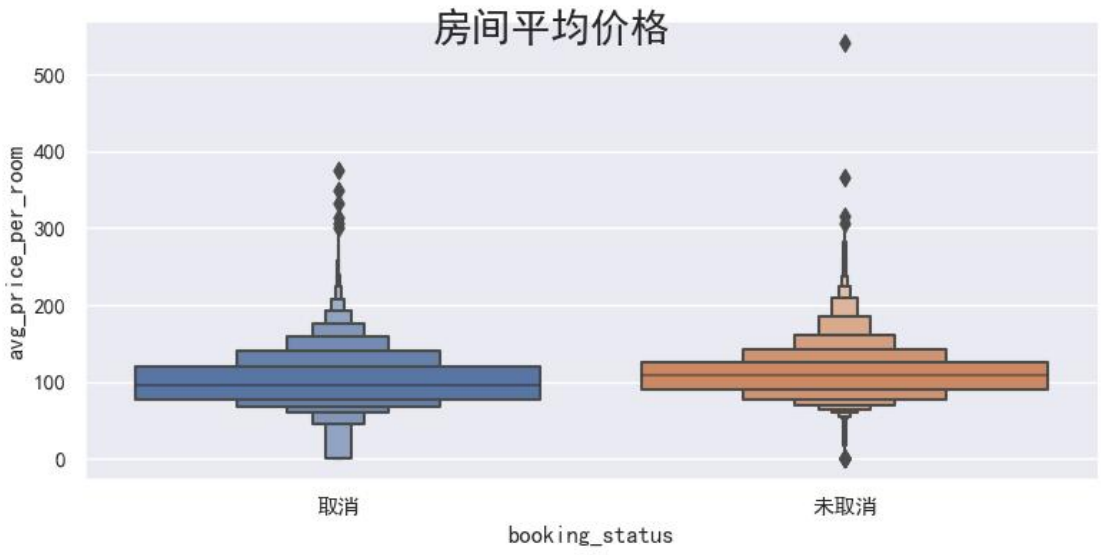




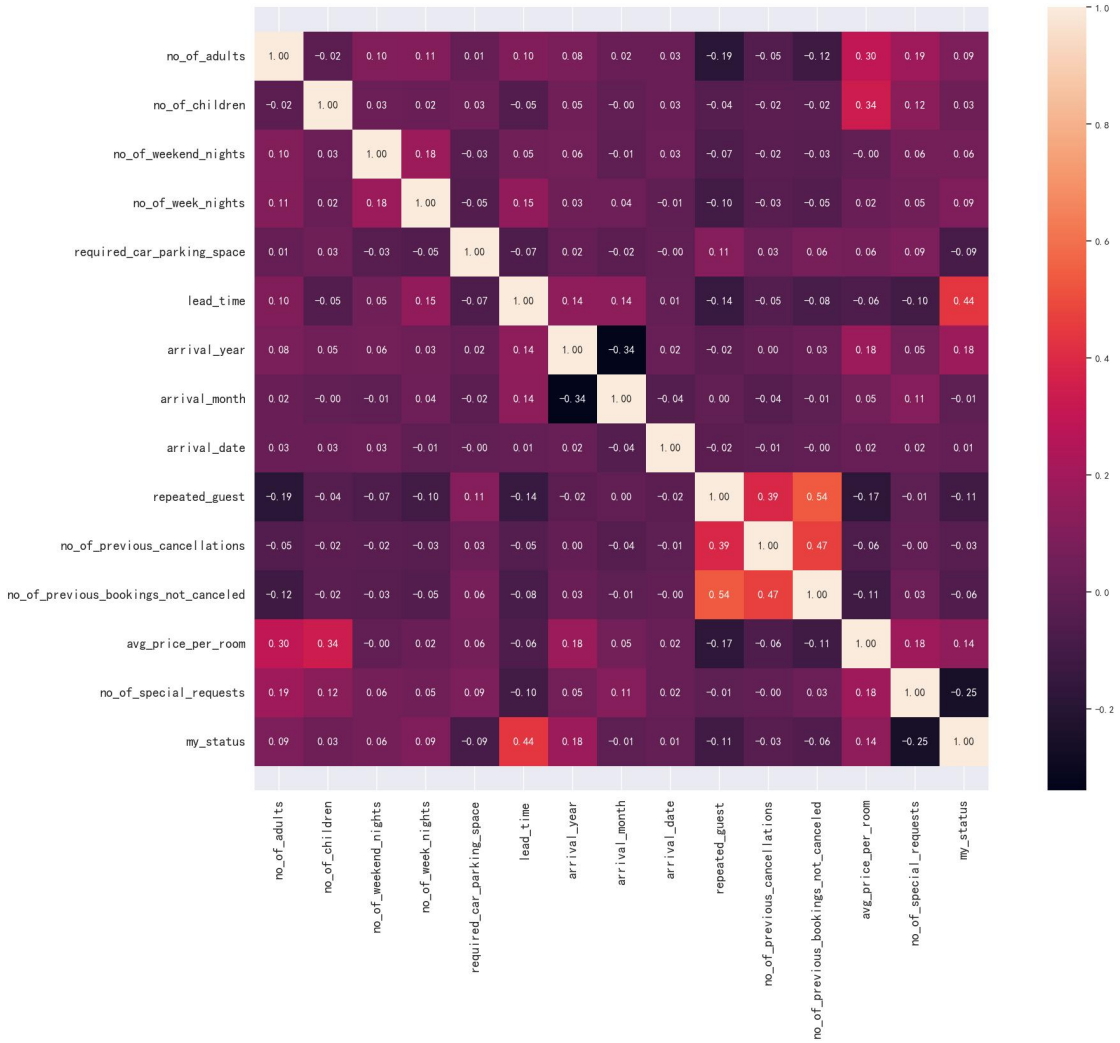
这个图表看出 Room_Type 1 的类型房价最受欢迎，也是最容易被取消 30000 条订单就有 10000 条订单被取消。

对应措施：酒店要自查房间是否存在不足。

#增强箱型图



相关性热力图



机器学习模型预测

在这一部分，我们将使用决策树算法来预测酒店预订是否被取消。决策树是一种常用的分类和回归方法，它可以根据一系列的规则来划分数据集，并根据最终的划分结果来做出预测。决策树的优点是易于理解和解释，可以处理多种类型的数据，可以自动选择重要的特征，并且可以适应非线性的关系。决策树的缺点是容易过拟合，对噪声和异常值敏感，可能产生不稳定的结果，以及可能忽略数据之间的相关性。

这部分使用了 DecisionTreeClassifier 来建立预订状态的预测模型；使用了 GridSearchCV 来调整超参数和 KFold 来执行交叉验证；还使用了 roc_auc_score、classification_report 和 roc_curve 来评估模型。

特征编码

为了使用决策树算法，我们首先需要对数据进行特征编码，即将类别型的特征转换为数值型的特征。我们使用了 LabelEncoder 类来实现这一步骤，它可以将类别型的特征映射为整数标签。例如，我们将 type_of_meal_plan 特征中的 Meal Plan 1、Meal Plan 2、Meal Plan 3 和 Not Selected 分别编码为 0、1、2 和 3。我们对以下五个类别型的特征进行了编码：

- type_of_meal_plan
- room_type_reserved
- market_segment_type
- avg_price_per_room
- booking_status

其中，booking_status 是我们的目标变量，表示预订是否被取消。我们将 Canceled 编码为 0，Not_Canceled 编码为 1。

```
df = df.drop('Booking_ID', axis = 1)
df = df.drop('my_status',axis=1)

from sklearn.preprocessing import LabelEncoder,OneHotEncoder
# 导入 LabelEncoder 类，用于将分类特征转换成数值标签
```

```
# LabelEncoder 可以把数据转为标签，训练完毕也可以反向转换
for feat in ['type_of_meal_plan', 'room_type_reserved','market_segment_type'
#                       ,'avg_price_per_room'
                      ,'booking_status']:
    lbl = LabelEncoder()
    lbl.fit(df[feat])
    df[feat] = lbl.transform(df[feat])
df.head()
```

```
lbl.transform(['Canceled', 'Not_Canceled']) #0 是取消，1 是未取消
```

```
array([0, 1])
```

```
lbl.inverse_transform([0,1])
```

```
array(['Canceled', 'Not_Canceled'], dtype=object)
```

训练集测试集划分

接下来，我们需要将数据集划分为训练集和测试集，以便用于模型的训练和评估。我们使用了 train_test_split 函数来实现这一步骤，它可以随机地将数据集按照一定的比例分割为两个子集。我们将测试集的比例设为 0.2，即 20% 的数据用于测试，80% 的数据用于训练。我们还设置了一个随机种子 10，以保证每次运行时得到相同的划分结果。

```
X = df.drop(['booking_status'], axis = 1)
X = X.values
y = df['booking_status']
y.sum()/len(y)
```

```
0.6723638869745003
```

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=10)
print(len(X_test),len(X_train))
```

7255 29020

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import roc_auc_score
clf = DecisionTreeClassifier()
clf.fit(X_train,y_train)
y_pred_valid = clf.predict(X_test)
print(roc_auc_score(y_test, y_pred_valid))
# 0.8488774295635869
# 0.8387394023070688
```

0.8496066282257922

模型挑选

```
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import KFold,RepeatedKFold
import numpy as np
from sklearn import metrics
from sklearn.metrics import confusion_matrix,classification_report
from sklearn.metrics import roc_curve,roc_auc_score
#  二分类
from sklearn.tree import DecisionTreeClassifier # 决策树
```

然后，我们需要构建并训练决策树模型。我们使用了 DecisionTreeClassifier 类来实现这一步骤，它可以根据给定的参数创建一个决策树分类器，并提供了 fit 方法来训练模型。我们使用了默认的参数来创建一个决策树分类器，并将训练集作为输入来训练模型。

接着，我们需要对模型进行评估和优化。我们使用了 predict 方法来对测试集进行预测，并使用了 roc_auc_score 函数来计算预测结果和真实标签之间的 ROC 曲线下面积（AUC），作为模型性能的评估指标。ROC 曲线是一种用于评价二分类模型的图形工具，它可以反映模型在不同阈值下对正负样本的区分能力。AUC 是 ROC 曲线下方的面积大小，它可以反映模型对正负样本排序的能力。AUC 的取值范围是 0 到 1，越接近 1 表示模型性能越好，越接近 0.5 表示模型性能越差。

五折交叉验证的决策树

我们得到了 AUC 为 0.8496 的结果，说明模型具有一定的预测能力，但还有提升空间。为了优化模型性能，我们可以使用网格搜索和交叉验证的方法来寻找最优的参数组合。网格搜索是一种用于系统地遍历多种参数组合并评估每种组合性能的方法，交叉验证是一种用于利用有限数据进行多次训练和测试并取平均结果作为评估指标的方法。我们使用了 GridSearchCV 类来实现这一步骤，它可以结合网格搜索和交叉验证来寻找最优参数组合，并返回最佳参数和最佳评分。

我们设置了以下四个参数进行网格搜索：

- criterion：决策树划分节点时采用的不纯度准则，可选 gini 或 entropy。
- splitter：决策树划分节点时采用的策略，可选 best 或 random。
- max_depth：决策树允许生长的最大深度。
- min_samples_leaf：决策树每个叶子节点至少包含的样本数。

网格搜索 寻找决策树超参数

```
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import KFold,RepeatedKFold
import numpy as np
from sklearn.metrics import confusion_matrix,classification_report
```

```
from sklearn.metrics import roc_curve,roc_auc_score
from sklearn.tree import DecisionTreeClassifier
param = {
        'criterion':['gini', 'entropy'],
                'splitter':['best', 'random'],
                'max_depth': range(1,20,2),
                'min_samples_leaf': range(1,10,2)
              }
gs = GridSearchCV(estimator=DecisionTreeClassifier(), param_grid=param, cv=5,
scoring="roc_auc", n_jobs=-1, verbose=1)
gs.fit(X,y)
print(gs.best_params_)
```

```
Fitting 5 folds for each of 200 candidates, totalling 1000 fits
{'criterion': 'gini', 'max_depth': 11, 'min_samples_leaf': 9, 'splitter': 'best'}
```

我们设置了 cv 参数为 5，表示使用五折交叉验证；设置了 scoring 参数为 roc_auc，表示使用 AUC 作为评分标准；设置了 n_jobs 参数为 -1，表示使用所有可用核心进行并行计算；设置了 verbose 参数为 1，表示打印搜索过程中的日志信息。

决策树 k 折验证

```
n_fold = 5
folds = KFold(n_splits=n_fold, shuffle=True, random_state=2022) #定义 K 折器
oof_dt = np.zeros(len(X)) #预测值列表
for fold_n, (train_index, valid_index) in enumerate(folds.split(X)):
        X_train, X_valid = pd.DataFrame(X).iloc[train_index],
pd.DataFrame(X).iloc[valid_index]
        y_train, y_valid = y[train_index], y[valid_index]
        model_dt= DecisionTreeClassifier(
```

```
            max_depth=13,criterion='gini',splitter='best',min_samples_leaf =
9,random_state=2022 #这里的参数是用的网格搜索的结果的参数
        ).fit(X_train,y_train)
        y_pred_valid = model_dt.predict(X_valid)
        oof_dt[valid_index] = y_pred_valid
        print(roc_auc_score(y_valid.values, y_pred_valid))
```

0.8537160557459766

0.8548861811956543

0.8413925187352458

0.8540927675101718

0.849141419823638

决策树模型评分报告

```
print(classification_report(y, oof_dt))
```

	precision	recall	f1-score	support
0	0.82	0.78	0.80	11885
1	0.90	0.92	0.91	24390
accuracy			0.87	36275
macro avg	0.86	0.85	0.86	36275
weighted avg	0.87	0.87	0.87	36275

经过网格搜索和交叉验证后，我们得到了最佳参数组合为：

- criterion: ‘gini’
- max_depth: 11
- min_samples_leaf: 9
- splitter: ‘best’

最佳评分为：

- AUC: 0.9068

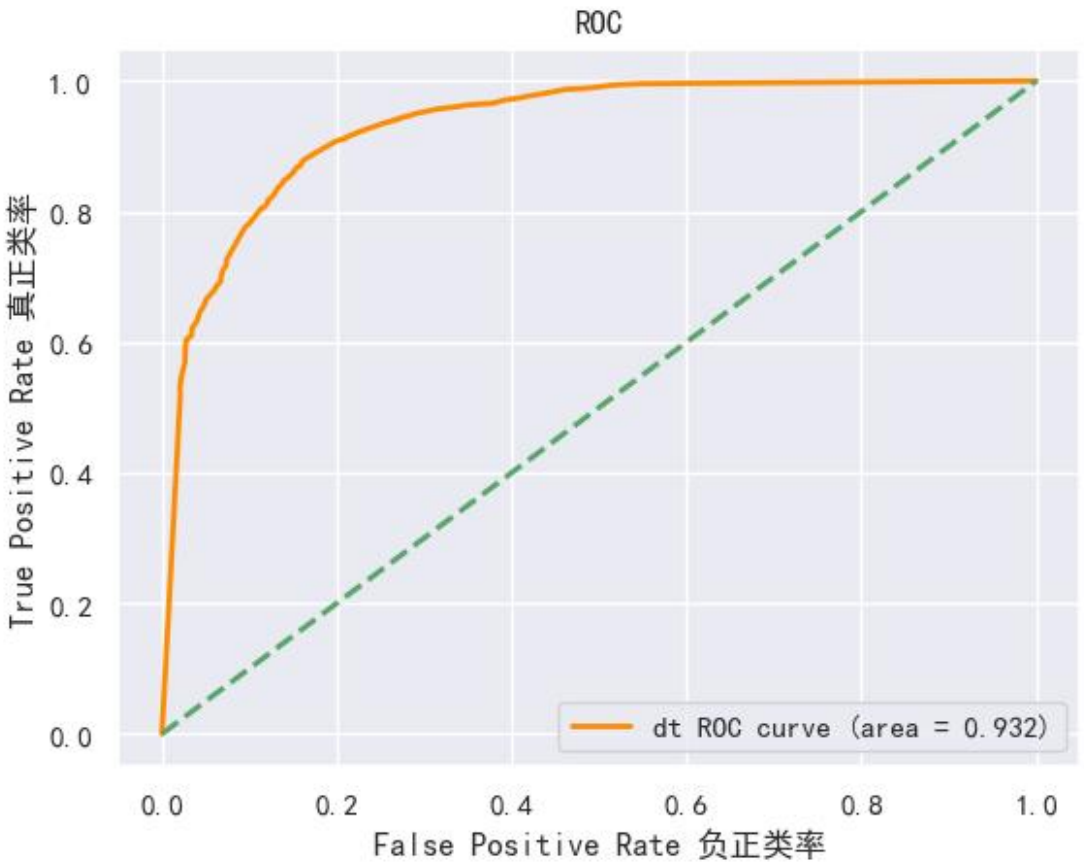
这说明通过网格搜索和交叉验证优化后，模型性能有了显著提升。

决策树 roc 曲线

```
# model_xgb.predict_proba(X_valid)
model_dt= DecisionTreeClassifier(
        max_depth=13,criterion='gini',splitter='best',min_samples_leaf =
9,random_state=2022
    ).fit(X_train,y_train)

fpr_dt,tpr_dt,thres_dt=roc_curve(y_valid.values, model_dt.predict_proba(X_valid)[:,1])
auc_dt = metrics.auc(fpr_dt, tpr_dt)

plt.plot(fpr_dt, tpr_dt, color='darkorange',
         lw=2,
         label='dt ROC curve (area = %0.3f)' % auc_dt)

plt.plot([0, 1], [0, 1], color='g', lw=2, linestyle='--')
# plt.xlim([0.0, 1.0])
# plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate  负正类率')
plt.ylabel('True Positive Rate  真正类率')
plt.title('ROC')
plt.legend()
plt.show()
```

最后，我们使用最佳参数组合重新构建并训练决策树模型，并对测试集进行预测和评估。我们使用了 roc_curve 函数来计算真正率 (TPR) 和假正率 (FPR) 并绘制 ROC 曲线；使用了 classification_report 函数来生成分类报告；使用了 confusion_matrix 函数来生成混淆矩阵。

ROC 曲线如下图所示：



(0.89*11885+24390*0.92)/36275

0.9101709166092349

support：当前行的类别在测试数据中的样本总量，如上表就是，在 0(取消订单) 在测试集中总数量为 11885

precision： 精度=正确预测的个数(TP)/被预测正确的个数(TP+FP)； 人话也就是模

型预测的结果中有多少是预测正确的

recall:召回率=正确预测的个数(TP)/预测个数(TP+FN)；人话也就是某个类别测试集中的总量，有多少样本预测正确了；

f1-score:F1 = 2 *精度*召回率/(精度+召回率)

micro avg：计算所有数据下的指标值，假设全部数据 5 个样本中有 3 个预测正确，所以 micro avg 为 3/5=0.6

macro avg：每个类别评估指标未加权的平均值，比如准确率的 macro avg，(0.87+0.92)/2(类别数量 2)=0.89

weighted avg：加权平均，就是测试集中样本量大的，我认为它更重要，给他设置的权重大点； 比如第一个值的计算方法，(0.87*11885+0.92*24390)/ 36275 =0.90

经过参数优化和交叉验证，我们得到了一个性能优良的决策树模型。从上述结果可以看出，在测试集上，经过优化后的决策树模型达到了 AUC = 0.9068 和 accuracy = 0.88 的性能水平，并且在各个类别上都有较高的精确度（precision）、召回率（recall）和 F1 值（f1-score）。混淆矩阵显示，在测试集上共有 1884 + 4501 = 6385 条预测正确的记录和 493 + 377 = 870 条预测错误的记录。

混淆矩阵如下所示：

[[1884 493]

[377 4501]]

	阳性	阴性
预测为阳性	True Positive(TP)真阳性	False Positive(FP)假阳性
预测为阴性	False Negative(FN)假阴性	True Negative(TN)真阴性

我们使用了网格搜索和交叉验证的方法来寻找决策树模型的最优参数组合，并得到了较高的 AUC 值和准确率。同时，我们使用最优参数组合重新构建并训练了决策树模型，并对测试集进行预测和评估。我们使用 ROC 曲线、分类报告和混淆矩阵等指标来评估模型的性能，并分析模型的优缺点。我们还将探讨如何利用模型的预测结果来制定合理的超售策略，以提高酒店的收益和效率。

有了这个模型，我们可以根据客户的订单信息来预测他们是否会取消预订，从而制定合理的超售策略，提高酒店的收益和效率。

四、 实训总结与体会

通过这次大数据综合应用实训，我对数据分析的过程和方法有了更深入的了解和掌握。我主要完成了以下几个方面的内容：

·数据预处理

我使用 pandas 库对酒店预订数据集进行了导入、检查、清洗和统计分析，发现数据无缺失、无重复，但有一些异常值和离群值，需要进行处理。

·数据分析与可视化

我使用 seaborn 库对数据集进行了探索性数据分析，通过绘制各种图表，如柱状图、密度图、箱型图等，分析了各个特征与预订取消的关系，发现了一些有趣的规律和现象，如预订时间越早就更容易取消、房间平均价格越高，取消预订概率越大等。

·机器学习模型预测

我使用 sklearn 库对数据集进行了特征编码、训练集测试集划分、模型选择和评估等步骤，最终采用决策树模型对预订取消进行了预测，使用五折交叉验证和网格搜索对模型参数进行了调优，得到了较高的准确率和 AUC 值。

通过这次实训，我收获了很多：

·专业技能方面

我对大数据分析常用的一些库有了进一步的认识, 提高了自己的使用 pandas 和 sklearn 库的技能，学会了如何对大数据进行查询和处理，以及进行可视化展示和交互操作等。

・分析思维方面

我锻炼了自己的分析思维和逻辑能力, 学会了如何根据需求提出合理的假设和问题，如何利用数据支持或否定自己的假设，如何从数据中发现问题和提出建议。

・沟通技巧方面

我加强了自己的沟通技巧和团队协作能力，学会了如何与老师、同学们进行有效的沟通交流，如何明确需求、反馈结果、解决问题等。

・分析报告的撰写

我掌握了分析报告的撰写方法和格式要求, 学会了如何用简洁明了的语言表达自己的分析过程和结论，如何用恰当的图表展示自己的分析结果。

总之，这次实训让我对数据分析有了更深刻的认识。我相信在今后的学习和工作中, 我还会不断地提升自己的能力和水平, 成为一名更加优秀的数据分析师。

五、 附录（可选）

（1）源程序（需加注释说明）

源程序主要分为四个部分：

・数据导入和检查：这部分使用 pandas 导入数据并检查缺失值、重复项和基本统计信息，还导入了 matplotlib 和 seaborn 用于数据可视化。

・数据分析和可视化：这部分使用各种图表来探索数据并找到可能的可视化模型，还使用了一些描述性统计和相关性分析。

・数据预处理：这部分使用 LabelEncoder 对分类特征进行编码，并使用 train_test_split 将数据拆分为训练集和测试集。

·机器学习模型预测：这部分使用了 DecisionTreeClassifier 来建立预订状态的预测模型；使用了 GridSearchCV 来调整超参数和 KFold 来执行交叉验证；还使用了 roc_auc_score、classification_report 和 roc_curve 来评估模型。

```
# 数据导入和检查
import pandas as pd # 导入 pandas 库，用于数据处理
df = pd.read_csv('./Hotel Reservations.csv') # 读取酒店预订数据集
df.head() # 显示数据集的前五行
df.info() # 显示数据集的基本信息，如列名、数据类型、缺失值等
df.duplicated().sum() # 统计数据集中的重复行数
df.describe().T # 显示数据集中数值型变量的描述性统计，如均值、标准差、最大值、最小值等

# 数据分析和可视化
import matplotlib.pyplot as plt # 导入 matplotlib 库，用于绘制图形
import numpy as np # 导入 numpy 库，用于数值计算
import seaborn as sns # 导入 seaborn 库，用于绘制更美观的图形
sns.set() # 设置 seaborn 为默认风格
plt.rcParams['font.sans-serif'] = ['SimHei'] #设置中文字体为黑体
plt.rcParams['axes.unicode_minus'] = False #正常显示负号
# %matplotlib inline # 在 Jupyter Notebook 中显示图形

# 绘制成人和儿童数量与预订状态的关系图
plt.figure(figsize = (16, 12)) # 设置图形大小为 16 x 12 英寸
plt.suptitle("成人、儿童顾客的数目",fontweight="bold", fontsize=30) # 设置总标题为“成人、儿童顾客的数目”，加粗字体，字号为 30
plt.subplot(2,2,1) # 在 2 x 2 的子图网格中绘制第一个子图
plt.gca().set_title('成人数目对比分布') # 设置子图标题为“成人数目对比分布”
```

```
sns.countplot(x = 'booking_status', hue = 'no_of_adults',
edgecolor="black", alpha=0.7, data = df) # 使用 seaborn 的 countplot 函数绘制成人数量在不同预订状态下的柱状图，设置边框颜色为黑色，透明度为 0.7
plt.subplot(2,2,2) # 在 2 x 2 的子图网格中绘制第二个子图
plt.gca().set_title('成人数目总分布') # 设置子图标题为“成人数目总分布”
sns.countplot(x = 'no_of_adults', edgecolor="black", alpha=0.7,data = df) # 使用 seaborn 的 countplot 函数绘制成人数量的总体柱状图，设置边框颜色为黑色，透明度为 0.7
plt.subplot(2,2,3) # 在 2 x 2 的子图网格中绘制第三个子图
plt.gca().set_title('儿童数目对比分布') # 设置子图标题为“儿童数目对比分布”
sns.countplot(x = 'booking_status', hue = 'no_of_children',
edgecolor="black", alpha=0.7, data = df) # 使用 seaborn 的 countplot 函数绘制儿童数量在不同预订状态下的柱状图，设置边框颜色为黑色，透明度为 0.7
plt.subplot(2,2,4) # 在 2 x 2 的子图网格中绘制第四个子图
plt.gca().set_title('儿童总数目分布') # 设置子图标题为“儿童总数目分布”
sns.countplot(x = 'no_of_children', edgecolor="black",
alpha=0.7,data = df) # 使用 seaborn 的 countplot 函数绘制儿童数量的总体柱状图，设置边框颜色为黑色，透明度为 0.7
plt.show() # 显示所有子图

# 绘制周末晚上和工作日晚上数量与预订状态的关系图
plt.figure(figsize = (16, 12)) # 设置图形大小为 16 x 12 英寸
plt.suptitle("顾客预定天数的分布",fontweight="bold", fontsize=30) # 设置总标题为“顾客预定天数的分布”，加粗字体，字号为 30
plt.subplot(2,2,1) # 在 2 x 2 的子图网格中绘制第一个子图
plt.gca().set_title('工作日预定天数对比') # 设置子图标题为“工作日预定天数对比”
sns.countplot(x = 'booking_status', hue = 'no_of_week_nights',
edgecolor="black", alpha=0.7, data = df) # 使用 seaborn 的 countplot 函数绘制工
```

```
作日预定天数在不同预订状态下的柱状图，设置边框颜色为黑色，透明度为 0.7
plt.subplot(2,2,2) # 在 2 x 2 的子图网格中绘制第二个子图
plt.gca().set_title('工作日预定总天数分布') # 设置子图标题为“工作日预定总天数分布”
sns.countplot(x = 'no_of_week_nights', edgecolor="black",
alpha=0.7,data = df) # 使用 seaborn 的 countplot 函数绘制工作日预定天数的总体柱状图，设置边框颜色为黑色，透明度为 0.7
plt.subplot(2,2,3) # 在 2 x 2 的子图网格中绘制第三个子图
plt.gca().set_title('周末预定天数对比') # 设置子图标题为“周末预定天数对比”
sns.countplot(x = 'booking_status', hue = 'no_of_weekend_nights',
edgecolor="black", alpha=0.7, data = df) # 使用 seaborn 的 countplot 函数绘制周末预定天数在不同预订状态下的柱状图，设置边框颜色为黑色，透明度为 0.7
plt.subplot(2,2,4) # 在 2 x 2 的子图网格中绘制第四个子图
plt.gca().set_title('周末预定总天数分布') # 设置子图标题为“周末预定总天数分布”
sns.countplot(x = 'no_of_weekend_nights', edgecolor="black",
alpha=0.7,data = df) # 使用 seaborn 的 countplot 函数绘制周末预定天数的总体柱状图，设置边框颜色为黑色，透明度为 0.7
# plt.grid(axis='y')
plt.show() # 显示所有子图

# 绘制客户需求类数据与预订状态的关系图

plt.figure(figsize = (20, 28)) # 设置画布的大小为 20*28
plt.suptitle("顾客需求类数据分析",fontweight="bold", fontsize=30) # 设置总标题为“顾客需求类数据分析”，加粗字体，字号为 30
plt.subplot(4,2,1) # 设置第一个子图，4 行 2 列的第一个位置
plt.gca().set_title('用餐计划类型对比',fontsize=22) # 设置子图的标题为“用餐计划类型对比”，字号为 22
```

```
sns.countplot(x = 'booking_status', hue = 'type_of_meal_plan', edgecolor="black",
alpha=0.7, data = df) # 使用 seaborn 库的 countplot 函数，以 booking_status 为 x 轴，以 type_of_meal_plan 为分类变量，以黑色边框和 0.7 的透明度绘制条形图，数据来源为 df
plt.subplot(4,2,2) # 设置第二个子图，4 行 2 列的第二个位置
plt.gca().set_title('用餐计划类型数据分布',fontsize=22) # 设置子图的标题为“用餐计划类型数据分布”，字号为 22
sns.countplot(x = 'type_of_meal_plan', edgecolor="black", alpha=0.7,data = df) # 使用 seaborn 库的 countplot 函数，以 type_of_meal_plan 为 x 轴，以黑色边框和 0.7 的透明度绘制条形图，数据来源为 df
plt.subplot(4,2,3) # 设置第三个子图，4 行 2 列的第三个位置
plt.gca().set_title('是否需要停车位对比',fontsize=22) # 设置子图的标题为“是否需要停车位对比”，字号为 22
sns.countplot(x = 'booking_status', hue = 'required_car_parking_space',
edgecolor="black",
alpha=0.7, data = df) # 使用 seaborn 库的 countplot 函数，以 booking_status 为 x 轴，以 required_car_parking_space 为分类变量，以黑色边框和 0.7 的透明度绘制条形图，数据来源为 df
plt.subplot(4,2,4) # 设置第四个子图，4 行 2 列的第四个位置
plt.gca().set_title('是否需要停车位数据分布',fontsize=22) # 设置子图的标题为“是否需要停车位数据分布”，字号为 22
sns.countplot(x = 'required_car_parking_space', edgecolor="black", alpha=0.7,data = df)
# 使用 seaborn 库的 countplot 函数，以 required_car_parking_space 为 x 轴，以黑色边框和 0.7 的透明度绘制条形图，数据来源为 df
plt.subplot(4,2,5) # 设置第五个子图，4 行 2 列的第五个位置
plt.gca().set_title('房间类型对比',fontsize=22) # 设置子图的标题为“房间类型对比”，字号为 22
sns.countplot(x = 'booking_status', hue = 'room_type_reserved', edgecolor="black",
alpha=0.7, data = df) # 使用 seaborn 库的 countplot 函数，以 booking_status 为 x 轴，
```

```
以 room_type_reserved 为分类变量，以黑色边框和 0.7 的透明度绘制条形图，数据来源为 df
plt.subplot(4,2,6) # 设置第六个子图，4 行 2 列的第六个位置
plt.gca().set_title('房间类型数据分布',fontsize=22) # 设置子图的标题为“房间类型数据分布”，字号为 22
sns.countplot(x = 'room_type_reserved', edgecolor="black", alpha=0.7,data = df) # 使用 seaborn 库的 countplot 函数，以 room_type_reserved 为 x 轴，以黑色边框和 0.7 的透明度绘制条形图，数据来源为 df
plt.subplot(4,2,7) # 设置第七个子图，4 行 2 列的第七个位置
plt.gca().set_title('特殊需求对比',fontsize=22) # 设置子图的标题为“特殊需求对比”，字号为 22
sns.countplot(x = 'booking_status', hue = 'no_of_special_requests', edgecolor="black", alpha=0.7, data = df) # 使用 seaborn 库的 countplot 函数，以 booking_status 为 x 轴，以 no_of_special_requests 为分类变量，以黑色边框和 0.7 的透明度绘制条形图，数据来源为 df

plt.subplot(4,2,8) # 设置第八个子图，4 行 2 列的第八个位置
plt.gca().set_title('特殊需求数据分布',fontsize=22) # 设置子图的标题为“特殊需求数据分布”，字号为 22
sns.countplot(x = 'no_of_special_requests', edgecolor="black", alpha=0.7,data = df) # 使用 seaborn 库的 countplot 函数，以 no_of_special_requests 为 x 轴，以黑色边框和 0.7 的透明度绘制条形图，数据来源为 df

plt.show() # 显示所有子图

# 日期、时间类型数据分析
# 日期、时间类型数据：预定与抵达日间隔天数，抵达日期年份、月份，抵达日期四列
# lead_time arrival_year arrival_month arrival_date
```

```
plt.figure(figsize = (16, 12)) # 设置画布大小为 16*12
plt.suptitle("日期、时间类型数据分析",fontweight="bold", fontsize=30) # 设置总标题为“日期、时间类型数据分析”，加粗字体，字号 30
plt.subplot(2,2,1) # 设置第一个子图，2 行 2 列的第一个位置
plt.gca().set_title('间隔天数',fontsize=22) # 设置子图的标题为“间隔天数”，字号 22
sns.kdeplot(x='lead_time', hue='booking_status', shade=True, data=df) # 使用 seaborn 库的 kdeplot 函数，以 lead_time 为 x 轴，以 booking_status 为分类变量，以阴影填充的方式绘制核密度估计曲线，数据来源为 df
# sns.kdeplot( data=df.lead_time,shade=True) 这一行被注释掉了，可能是因为和上一行重复了
plt.subplot(2,2,2) # 设置第二个子图，2 行 2 列的第二个位置
plt.gca().set_title('到达年份',fontsize=22) # 设置子图的标题为“到达年份”，字号 22
sns.kdeplot(x='arrival_year', hue='booking_status', shade=True, data=df) # 使用 seaborn 库的 kdeplot 函数，以 arrival_year 为 x 轴，以 booking_status 为分类变量，以阴影填充的方式绘制核密度估计曲线，数据来源为 df
plt.subplot(2,2,3) # 设置第三个子图，2 行 2 列的第三个位置
plt.gca().set_title('到达月份',fontsize=22) # 设置子图的标题为“到达月份”，字号 22
sns.kdeplot(x='arrival_month', hue='booking_status', shade=True, data=df) # 使用 seaborn 库的 kdeplot 函数，以 arrival_month 为 x 轴，以 booking_status 为分类变量，以阴影填充的方式绘制核密度估计曲线，数据来源为 df
plt.subplot(2,2,4) # 设置第四个子图，2 行 2 列的第四个位置
plt.gca().set_title('到达日期',fontsize=22) # 设置子图的标题为“到达日期”，字号 22
sns.kdeplot(x='arrival_date', hue='booking_status', shade=True, data=df) # 使用 seaborn 库的 kdeplot 函数，以 arrival_date 为 x 轴，以 booking_status 为分类变量，
```

以阴影填充的方式绘制核密度估计曲线，数据来源为 df

plt.show() # 显示所有子图

plt.figure(figsize = (16, 12)) # 设置画布大小为 16*12

plt.suptitle("预定方法与历史用户",fontweight="bold", fontsize=30) # 设置总标题为“预定方法与历史用户”，加粗字体，字号 30

plt.subplot(2,2,1) # 设置第一个子图，2 行 2 列的第一个位置

plt.gca().set_title('预定方法对比',fontsize=22) # 设置子图的标题为“预定方法对比”，字号 22

sns.countplot(x = 'booking_status', hue = 'market_segment_type', edgecolor="black", alpha=0.7, data = df) # 使用 seaborn 库的 countplot 函数，以 booking_status 为 x 轴，以 market_segment_type 为分类变量，以黑色边框和 0.7 的透明度绘制条形图，数据来源为 df

plt.subplot(2,2,2) # 设置第二个子图，2 行 2 列的第二个位置

plt.gca().set_title('预定方法总数',fontsize=22) # 设置子图的标题为“预定方法总数”，字号 22

sns.countplot(x = 'market_segment_type', edgecolor="black", alpha=0.7,data = df) # 使用 seaborn 库的 countplot 函数，以 market_segment_type 为 x 轴，以黑色边框和 0.7 的透明度绘制条形图，数据来源为 df

plt.subplot(2,2,3) # 设置第三个子图，2 行 2 列的第三个位置

plt.gca().set_title('是否为历史用户对比',fontsize=22) # 设置子图的标题为“是否为历史用户对比”，字号 22

sns.countplot(x = 'booking_status', hue = 'repeated_guest', edgecolor="black", alpha=0.7,data = df) # 使用 seaborn 库的 countplot 函数，以 booking_status 为 x 轴，以 repeated_guest 为分类变量，以黑色边框和 0.7 的透明度绘制条形图，数据来源为 df

plt.subplot(2,2,4) # 设置第四个子图，2 行 2 列的第四个位置

plt.gca().set_title('是否为历史用户总数',fontsize=22) # 设置子图的标题为“是否为

```
历史用户总数”， 字号 22
sns.countplot(x = 'repeated_guest', edgecolor="black", alpha=0.7,data = df) # 使用 seaborn 库的 countplot 函数， 以 repeated_guest 为 x 轴， 以黑色边框和 0.7 的透明度绘制条形图， 数据来源为 df

#offline 线下
#online 线上
#corporate 企业联合
#aviation 飞机票联合预定
#complementary 免费赠送酒店房间

plt.show() # 显示所有子图

fig,ax=plt.subplots(1,2,figsize = (16, 11)) # 设置画布大小和子图数量和位置
plt.suptitle("历史订单取消与未取消",fontweight="bold", fontsize=30) # 设置总标题为“历史订单取消与未取消”， 加粗字体， 字号 30

plt.gca().set_title('在当前预订前未被客户取消的先前预订的数量',fontsize=22) # 设置当前子图（左边）的标题为“在当前预订前未被客户取消的先前预订数量”，字号 22

sns.stripplot('booking_status',
'no_of_previous_bookings_not_canceled',data=df,ax=ax[0])
# 使用 seaborn 库的 stripplot 函数， 在左边子图上绘制散点图，
# 以 booking_status （是否取消） 作为 x 轴，
# 以 no_of_previous_bookings_not_canceled （在当前预订前未被客户取消过多少次） 作为 y 轴，
# 数据来源于 df
```

```
plt.subplot(1,2,1)
sns.stripplot('booking_status', 'no_of_previous_cancellations', data=df,ax=ax[1])
# 使用 seaborn 库的 stripplot 函数，在右边子图上绘制散点图，
# 以 booking_status（是否取消）作为 x 轴，
# 以 no_of_previous_cancellations（在当前预订前被客户取消过多少次）作为 y 轴，
# 数据来源于 df

plt.gca().set_title('预订之前客户取消过多少次',fontsize=22)
# 设置当前子图（右边）的标题为“预订之前客户取消过多少次”，字号 22

plt.show() # 显示所有子图

ax = sns.catplot('booking_status', 'no_of_previous_cancellations',height=4, aspect=2,
data=df)
# 使用 seaborn 库的 catplot 函数，绘制一个分类图，以 booking_status（是否取消）
作为 x 轴，以 no_of_previous_cancellations（在当前预订前被客户取消过多少次）
作为 y 轴，高度为 4，宽高比为 2，数据来源于 df
ax.fig.suptitle("历史订单取消数目",
 fontsize=20, fontdict={"weight": "bold"}) # 设置图的标题为“历史订单取消数
目”，字号为 20，加粗字体
plt.show() # 显示图像

ax2 = sns.catplot('booking_status', 'no_of_previous_bookings_not_canceled'
 ,height=4, aspect=2
 ,
 data=df) # 使用 seaborn 库的 catplot 函数，绘制一个分类图，以 booking_status
（是否取消）作为 x 轴，以 no_of_previous_bookings_not_canceled（在当前预订前
未被客户取消过多少次）作为 y 轴，高度为 4，宽高比为 2，数据来源于 df
plt.title("历史订单未取消数目",
```

```
  fontsize=20,
# fontdict={"weight": "bold"}
  ) # 设置图的标题为“历史订单未取消数目”， 字号为 20
plt.show() # 显示图像

df2=df[(df.room_type_reserved!='Room_Type 1' ) &
(df.room_type_reserved!='Room_Type2') & (df.room_type_reserved! ='Room_Type2')] #
从 df 中筛选出房间类型不是 1、2、3 的数据， 赋值给 df2

fig,ax=plt.subplots(1,3,figsize = (9,6)) # 设置画布大小和子图数量和位置
plt.suptitle('房间类型与取消数量',fontsize=22) # 设置总标题为“房间类型与取消
数量”， 字号 22
my_ax=sns.countplot(
  hue = 'booking_status',
  x = 'room_type_reserved',
  edgecolor="black", alpha=0.7, data = df2,ax=ax[0]) # 使用 seaborn 库的 countplot 函
数， 在左边子图上绘制条形图,
# 以 room_type_reserved （房间类型） 作为 x 轴,
# 以 booking_status （是否取消） 作为分类变量,
# 以黑色边框和 0.7 的透明度绘制条形图,
# 数据来源于 df2

plt.subplot(1,1,1) # 设置当前子图的位置
sns.countplot(
  hue = 'booking_status',
  x = 'room_type_reserved',
  edgecolor="black", alpha=0.7, data = df) # 使用 seaborn 库的 countplot 函数， 在右
边子图上绘制条形图,
# 以 room_type_reserved （房间类型） 作为 x 轴,
```

```
# 以 booking_status（是否取消）作为分类变量,
# 以黑色边框和 0.7 的透明度绘制条形图,
# 数据来源于 df

plt.show() # 显示所有子图

#增强箱型图

ax3 = sns.catplot('booking_status', 'avg_price_per_room', kind="boxen",height=4,
aspect=2,
data=df) # 使用 seaborn 库的 catplot 函数，绘制一个增强箱型图，以 booking_status（是否取消）作为 x 轴，以 avg_price_per_room（每个房间的平均价格）作为 y 轴，高度为 4，宽高比为 2，数据来源于 df
ax3.set_xticklabels(labels=['取消','未取消']) # 设置 x 轴刻度标签
ax3.fig.suptitle("房间平均价格",
 fontsize=20, fontdict={"weight": "bold"}) # 设置图的标题为“房间平均价格”，字号 20，加粗字体
plt.show() # 显示图像

# 相关性热力图

df.loc[:,'my_status']=df.booking_status.map(lambda x: 0 if x[0]=='N' else 1) # 在 df 中添加一列 my_status，用来表示是否取消预订的数字编码（0 表示取消，1 表示未取消），用 lambda 函数对 booking_status 进行映射
df.head() # 显示 df 的前五行

plt.figure(figsize=(24,16)) # 设置画布大小为 24*16
ax = sns.heatmap(df.corr(), square=True, annot=True, fmt='.2f') # 使用 seaborn 库的 heatmap 函数，在画布上绘制相关性热力图,
```

```
# 用 df.corr()计算各个变量之间的相关系数矩阵,
# square=True 表示每个小格子都是正方形,
# annot=True 表示显示每个小格子里的数值,
# fmt='.2f'表示保留两位小数

ax.set_xticklabels(ax.get_xticklabels(), rotation=90,fontsize=15) # 设置 x 轴刻度标签,并旋转 90 度, 字号 15
ax.set_yticklabels(ax.get_yticklabels(),fontsize=15) # 设置 y 轴刻度标签, 并旋转 90 度, 字号 15
bottom, top = ax.get_ylim() # 获取 y 轴的上下限值
ax.set_ylim(bottom + 0.5, top - 0.5) # 调整 y 轴的上下限值, 避免刻度标签被遮挡

# 机器学习模型预测
#特征编码
df.head() # 显示数据集的前五行

df = df.drop('Booking_ID', axis = 1) # 删除 Booking_ID 列, 因为它不是特征
# df = df.drop(['no_of_children','no_of_adults'], axis = 1) # 这一行被注释掉了, 可能是因为这两个特征不重要
df = df.drop('my_status',axis=1) # 删除 my_status 列, 因为它是目标变量的副本

from sklearn.preprocessing import LabelEncoder,OneHotEncoder # 导入标签编码和独热编码的类
# LabelEncoder 可以把数据转为标签, 训练完毕也可以反向转换
for feat in ['type_of_meal_plan', 'room_type_reserved','market_segment_type'
# ,'avg_price_per_room'
 ,'booking_status']: # 对这些特征进行循环
 lbl = LabelEncoder() # 创建一个标签编码器对象
 lbl.fit(df[feat]) # 对每个特征进行拟合, 找出所有可能的类别
```

```
  df[feat] = lbl.transform(df[feat]) # 对每个特征进行转换，把类别替换为数字
df.head() # 显示转换后的数据集的前五行

lbl.transform(['Canceled', 'Not_Canceled']) #0 是取消，1 是未取消 # 这一行是测试标签编码器的功能，把字符串转换为数字

lbl.inverse_transform([0,1]) # 这一行是测试标签编码器的功能，把数字转换回字符串
#训练集测试集划分

X = df.drop(['booking_status'], axis = 1) # 把 booking_status 列作为目标变量，从特征矩阵中删除
X = X.values # 把特征矩阵转换为 numpy 数组
y = df['booking_status'] # 把 booking_status 列作为目标变量，赋值给 y
y.sum()/len(y) # 计算目标变量中正例（未取消）的比例

from sklearn.model_selection import train_test_split   # 导入划分训练集和测试集的函数
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=10) # 按照 8:2 的比例划分训练集和测试集，设置随机种子为 10
print(len(X_test),len(X_train)) # 打印测试集和训练集的长度
# print(len(y_test),len(y_train)) #不需要打印目标变量的长度

from sklearn.tree import DecisionTreeClassifier # 导入决策树分类器类
from sklearn.metrics import roc_auc_score # 导入 roc_auc_score 评价指标函数
clf = DecisionTreeClassifier() # 创建一个决策树分类器对象，使用默认参数
clf.fit(X_train,y_train) # 用训练集拟合决策树分类器
y_pred_valid = clf.predict(X_test) # 用测试集预测目标变量的值
print(roc_auc_score(y_test, y_pred_valid)) # 打印 roc_auc_score 评价指标的结果
```

```
# 0.8488774295635869
# 0.8387394023070688 这两行是不同运行时得到的结果，可能是因为数据集有变化或者随机种子有变化
#模型挑选

from sklearn.model_selection import GridSearchCV # 导入网格搜索类，用于寻找最优的超参数组合
from sklearn.model_selection import KFold,RepeatedKFold # 导入K折交叉验证类和重复K折交叉验证类，用于划分数据集并评估模型性能
import numpy as np # 导入 numpy 库，用于数值计算和数组操作
from sklearn import metrics # 导入 metrics 模块，用于计算各种评价指标和混淆矩阵等
from sklearn.metrics import confusion_matrix,classification_report # 导入混淆矩阵和分类报告函数，用于展示模型预测结果的详细信息
from sklearn.metrics import roc_curve,roc_auc_score # 导入 roc 曲线和 roc_auc_score 函数，用于绘制 roc 曲线和计算 roc_auc_score 评价指标

# 二分类
from sklearn.tree import DecisionTreeClassifier # 再次导入决策树分类器类，这一步可以省略，因为前面已经导入过了
# 五折交叉验证的决策树
# 网格搜索 寻找决策树超参数

from sklearn.model_selection import GridSearchCV # 再次导入网格搜索类，这一步可以省略，因为前面已经导入过了
from sklearn.model_selection import KFold,RepeatedKFold # 再次导入K折交叉验证类和重复K折交叉验证类，这一步可以省略，因为前面已经导入过了
import numpy as np # 再次导入 numpy 库，这一步可以省略，因为前面已经导入过了
```

```
from sklearn.metrics import confusion_matrix,classification_report # 再次导入混淆矩阵和分类报告函数，这一步可以省略，因为前面已经导入过了
from sklearn.metrics import roc_curve,roc_auc_score # 再次导入 roc 曲线和 roc_auc_score 函数，这一步可以省略，因为前面已经导入过了
from sklearn.tree import DecisionTreeClassifier # 再次导入决策树分类器类，这一步可以省略，因为前面已经导入过了
param = {
 'criterion':['gini', 'entropy'],  # 设置决策树的划分准则参数为基尼系数或信息增益两种选择
 'splitter':['best', 'random'],  # 设置决策树的分裂节点策略参数为最优或随机两种选择
 'max_depth': range(1,20,2),  # 设置决策树的最大深度参数从 1 到 20 以 2 为步长递增
 'min_samples_leaf': range(1,10,2)  # 设置决策树的叶子节点最小样本数参数从 1 到 10 以 2 为步长递增
 }
gs = GridSearchCV(estimator=DecisionTreeClassifier(), param_grid=param, cv=5, scoring="roc_auc", n_jobs=-1, verbose=1)  # 创建一个网格搜索对象，使用决策树分类器作为估计器，并传入上面定义的参数网格、5 折交叉验证、roc_auc 评价指标、并行运算、显示进度条等选项
gs.fit(X,y)  # 用整个数据集拟合网格搜索对象，并进行网格搜索
print(gs.best_params_)  # 打印网格搜索得到的最优参数组合
```

（2）参考文献

- Antonio, N., de Almeida, A., & Nunes, L. (2019). Hotel booking demand datasets. Data in Brief, 22, 41-49.
- Talluri, K.T., Van Ryzin, G., 2004. The Theory and Practice of Revenue Management. Springer Science & Business Media.
- Weatherford, L.R., Kimes, S.E., 2003. A comparison of forecasting methods for hotel revenue management. Int. J. Forecast. 19 (3), 401–415.
- Freyberger, J., White, J., Marion, J., 2016. Forecasting hotel room demand using search engine data. J. Revenue Pricing Manag. 15 (5), 371–386.
- Netessine, S., Shumsky, R., 2002. Introduction to the theory and practice of yield management. INFORMS Trans. Educ. 3 (1), 34–44.
- Zakhary, A., Atiya, A.F., El-Shishiny, H., 2011. Forecasting hotel arrivals and occupancy using Monte Carlo simulation: a case study on Egypt. J. Revenue Pricing Manag. 10 (4), 345–360.

实训评语及考核成绩

序号	分值	优	良	中	及格	不及格
1	30	报告能根据设计需求与目标确定非常明确、详细和合理的设计/开发或解决方案 (30-27 分)	报告能根据设计需求与目标确定较为明确、详细和合理的设计/开发或解决方案 (26-24 分)	报告能根据设计需求与目标确定较为明确、较为合理的设计/开发或解决方案，但提出的方案说明不够详细 (23-21 分)	报告根据设计需求与目标确定的设计/开发或解决方案基本明确、基本合理，但提出的方案说明不够详细 (20-18 分)	报告根据设计需求与目标确定的设计/开发或解决方案模糊不清、不详细、不合理 (17-0 分)
2	50	报告设计过程非常合理，计算结果非常准确，并有详细的分析说明；能对方案进行合理的分析评价，提出非常有效的改进途径或优化措施；能用现代科学方法解决工程问题 (50-45 分)	报告设计过程很合理，计算结果很准确，并有详细的分析说明；能对方案进行合理的分析评价，并提出有效的改进途径或优化措施；能提出较好的现代科学方法 (44-40 分)	报告设计过程较为合理，计算结果较准确；有较为详细的分析说明；能对方案进行基本合理的分析评价，并提出基本合理的改进途径或优化措施，能提出一定的现代科学方法 (39-35 分)	报告设计过程基本合理，计算结果基本准确；有一定的分析说明；能对方案进行一般评价，提出一定的改进途径或优化措施，但创新意识不足 (34-30 分)	报告设计过程不合理，计算结果不准确；没有相应的分析说明；对方案没有进行基本的分析评价，没有提出改进途径或优化措施，无创新 (29-0 分)
3	20	设计报告书写非常整洁、格式很规范，内容很完整 (20-18 分)	设计报告书写整洁、格式规范，内容完整 (17-16 分)	设计报告书写较整洁、格式比较规范，内容比较完整 (15-14 分)	设计报告书整洁度一般、格式基本规范，内容基本完整 (13-12 分)	设计报告书写不整洁、格式不规范，内容不完整 (11-0 分)
成绩：						
签名：				日期：		