数据挖掘互评作业三
分类、预测与聚类

学　　院：	计算机学院
专　　业：	软件工程
指 导 教 师：	汤世平
姓　　名：	田效宇
学　　号：	3220201086

2021 年 06 月 05 日

1 所选数据集

名称：Hotel booking demand

描述：该数据集包含城市酒店和度假酒店的预订信息，包括预订时间、停留时间，成人/儿童/婴儿人数以及可用停车位数量等信息。

数据量：32 列，共 12W 条

分析：

（1）基本情况：城市酒店和假日酒店预订需求和入住率比较

（2）用户行为：提前预订时间、入住时长、预订间隔、餐食预订情况

（3）一年中最佳预订酒店时间

（4）利用 Logistic 预测酒店预订

2 数据预处理

如图所示：

```
In [26]: hotel = pd.read_csv('D:/hotel_bookings.csv')
         hotel
```

Out[26]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_number	arrival_date_day_of_month	stays_in_weekend_nights	s
0	Resort Hotel	0	342	2015	July	27	1	0	
1	Resort Hotel	0	737	2015	July	27	1	0	
2	Resort Hotel	0	7	2015	July	27	1	0	
3	Resort Hotel	0	13	2015	July	27	1	0	
4	Resort Hotel	0	14	2015	July	27	1	0	
...	...	...	...	...	...	...	...	...	
119385	City Hotel	0	23	2017	August	35	30	2	
119386	City Hotel	0	102	2017	August	35	31	2	
119387	City Hotel	0	34	2017	August	35	31	2	
119388	City Hotel	0	109	2017	August	35	31	2	
119389	City Hotel	0	205	2017	August	35	29	2	

119390 rows × 32 columns

Hotel booking demand 数据集的主要属性包括：

is_canceled:标志这间房间是否已经被取消。

lead_time:预订日期和到达日期之间经过的天数

arrival_date_year:到达的年份

arrival_date_month：到达的月份

arrival_date_week_number：到达的星期

arrival_date_day_of_month：到达的日期

stays_in_weekend_nights：客人在周末预订或留宿的天数

stays_in_week_nights：客人在周中预订或留宿的天数

… …

缺失值处理，利用出现最频繁的数据来填补缺失值，如图所示：

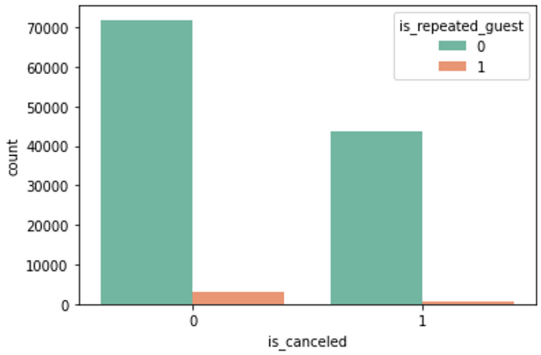
```
hotel.isnull().sum()
hotel.fillna(hotel.mode().iloc[0], inplace=True)
hotel.isnull().sum()
```

3 基本数据分析

首先观察入住和取消情况，如图所示：

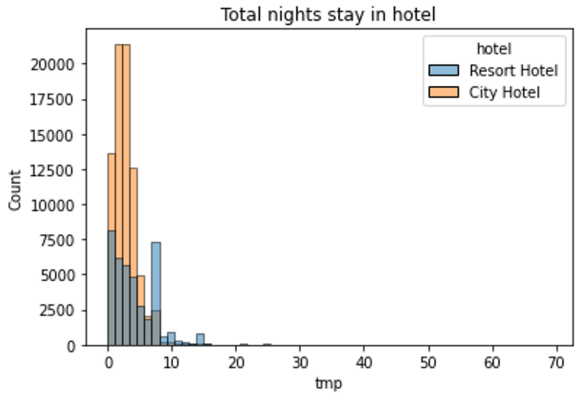
新老客户入住和取消占比：可以看出酒店的老用户数量相较于新用户的数量较少，而且新用户取消预订的比例较高，老用户的取消比例很低。

```
sns.countplot(x='is_canceled',hue='is_repeated_guest',data=hotel,palette='Set2')
plt.show()
```

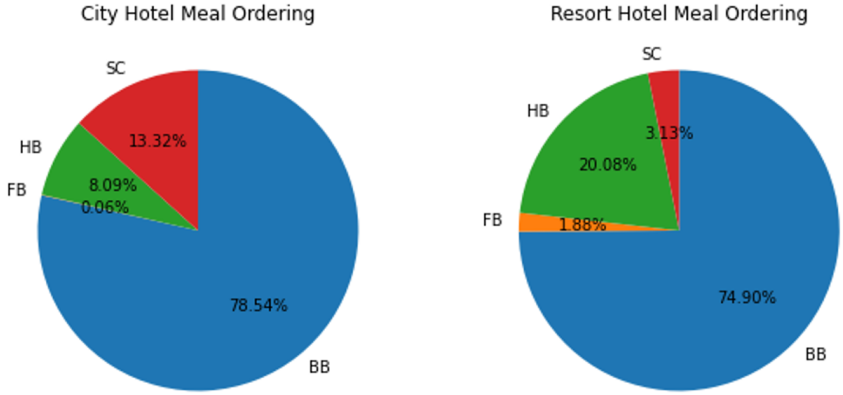


用户入住时长分析：城市酒店和度假酒店入住客人的入住时长一般都不会超过 10 天，城市酒店大部分客户的居住时间为 1-4 天，而度假酒店的客户的居住时间大部分为 1 天或 7 天。

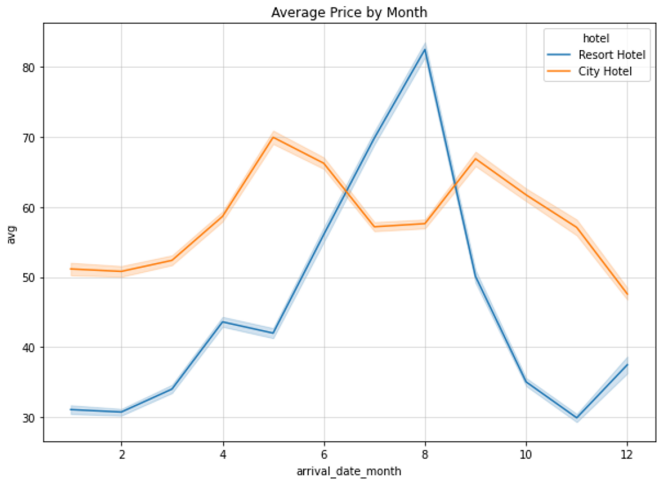
```
hotel['tmp'] = hotel['stays_in_weekend_nights']+hotel['stays_in_week_nights']
sns.histplot(x='tmp',hue='hotel',data=hotel,bins=60)
plt.title('Total nights stay in hotel')
plt.show()
```



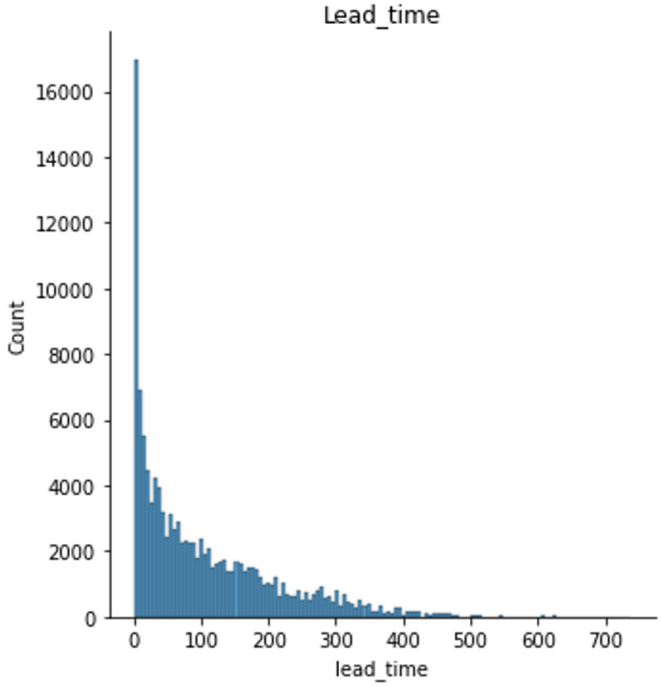
订餐情况：大部分用户都会订餐，其中订早餐所占比例最高。而度假酒店的订餐比例要超过城市酒店，三餐都订的比例很少，可能是因为中午客户一般不会回酒店就餐。



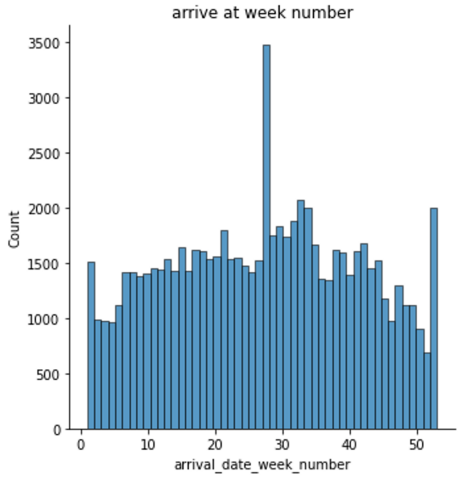
人均花销分布：城市酒店的价格在夏季降低，而度假酒店的价格会在夏季显著升高，甚至接近城市酒店同期价格的两倍，可能由于夏季人们倾向于去度假放松。城市酒店的价格分布比较均匀，而度假酒店价格起伏很大，推荐人少钱少的淡季去旅行。



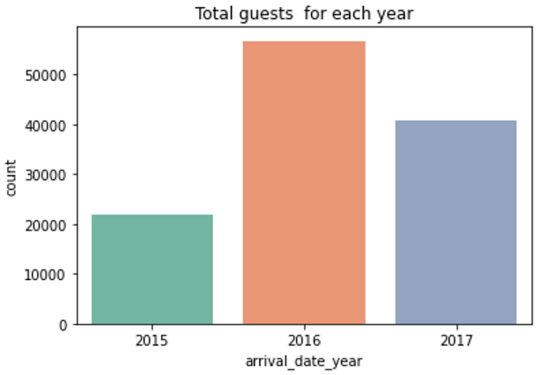
客户提前预定情况分布：大量的用户会选择提前 1-2 天预定，几乎占据所有客户数量的一半，但仍然有不少的用户会提前超过一年来进行预定。



客户到达的周数分布：一年之内每周客户到达的数量总体稳定，7-8 月份到达的用户数较多，而 1-2 月份到达的用户数较少。



每年客户总数分布：可以看出 2016 年的用户总量最多，甚至达到了前一年用户总量的 2 倍以上，而 2017 年的用户总量有所下降。



每月客户数量分布：可以看出每月的用户数量大体分布趋势和每周的分布情况类似，7-9 月份的人数明显上升，同时全年城市酒店的预定数量都要超过度假酒店。

4 利用逻辑回归预测酒店预订

首先去除与预测无关的几个属性:agent、company、hotel 以及reservation_status_data,is_canceled 属性反应了客户最终是否取消了预定，因此可以将该属性看作特征属性，划分特征和标签：

```
X = hotel.iloc[:,1:]
y = hotel.iloc[:,0]
```

其中 X 作为特征，Y 作为标签。查看 X 的大体分布：

```
RangeIndex: 119390 entries, 0 to 119389
Data columns (total 27 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   lead_time                       119390 non-null  int64
 1   arrival_date_year               119390 non-null  int64
 2   arrival_date_month              119390 non-null  object
 3   arrival_date_week_number        119390 non-null  int64
 4   arrival_date_day_of_month       119390 non-null  int64
 5   stays_in_weekend_nights         119390 non-null  int64
 6   stays_in_week_nights            119390 non-null  int64
 7   adults                          119390 non-null  int64
 8   children                        119390 non-null  float64
 9   babies                          119390 non-null  int64
 10  meal                            119390 non-null  object
 11  country                         119390 non-null  object
 12  market_segment                  119390 non-null  object
 13  distribution_channel            119390 non-null  object
 14  is_repeated_guest               119390 non-null  int64
 15  previous_cancellations          119390 non-null  int64
 16  previous_bookings_not_canceled  119390 non-null  int64
 17  reserved_room_type              119390 non-null  object
 18  assigned_room_type              119390 non-null  object
 19  booking_changes                 119390 non-null  int64
 20  deposit_type                    119390 non-null  object
 21  days_in_waiting_list            119390 non-null  int64
 22  customer_type                   119390 non-null  object
 23  adr                             119390 non-null  float64
 24  required_car_parking_spaces     119390 non-null  int64
 25  total_of_special_requests       119390 non-null  int64
 26  reservation_status              119390 non-null  object
```

由于逻辑回归算法只能处理数值特征，而数据集中的特征既包含数值特征又包含非数值特征，所以使用 python 的 sklearn 库中的 OneHotEncoder 函数将非数值特征转化为数值特征再进行后续的处理：

```
In [144]: need_transfer=['meal','distribution_channel','reservation_status','country','arrival_date_month','market_segment',
          'deposit_type','customer_type', 'reserved_room_type','assigned_room_type' ]
          transfer = make_column_transformer(
              (OneHotEncoder(),need_transfer), remainder = 'passthrough'
              )
          X = transfer.fit_transform(X).toarray()
```

划分数据集，将 80%的数据作为训练集，20%的数据作为测试集：

```
In [145]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
          X_train.shape
Out[145]: (95512, 256)
```

对于非数值特征要进行进一步的处理，将非数值特征转化为数值特征后数据的维数增加至了 256，需要进行降维。将数据集进行标准化后利用 python 的 PCA 库进行降维，将数据维数降低至 100。

```
pca = PCA(n_components = 100)
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)
explained_variance = pca.explained_variance_ratio_
print("降维后的训练数据:",X_train.shape,"\n测试数据:",X_test.shape)
print("降维后的训练数据:",X_train,"\n测试数据:",X_test)
```

```
降维后的训练数据: (95512, 100)
测试数据: (23878, 100)
降维后的训练数据: [[-2.60104561  2.72941406  1.57171688 ...  6.13857213  1.33389716
   2.81339735]
 [-2.08000317  2.27406685  1.2287991  ... -1.60547406  3.63237657
  -2.09493051]
 [-1.40726114  0.10930381 -1.37663922 ...  0.59061836  0.39505608
  -0.312809  ]
 ...
 [-2.67197437  0.1559445   0.06262923 ...  1.43613012 -0.44108899
  -0.53075844]
 [ 1.46754423 -0.64442668 -2.50387854 ... -0.36532173  0.28767086
  -0.70277379]
 [ 3.87692566  1.20172115  2.8114168  ...  0.07406237 -0.13698159
   0.013139  ]]
测试数据: [[ 0.84557478  0.66848904 -1.39546735 ...  0.7902706  -0.01617924
   0.15908805]
 [-0.33360933  0.87905479 -1.33906965 ... -0.23223144  0.09873799
  -0.41306128]
 [ 5.29041186  0.68283592  1.47537439 ... -0.2369942   0.19030562
  -0.20357923]
 ...
 [ 1.02160122 -4.44724511 -1.74792499 ... -0.09841479  0.42820951
  -1.2337255 ]
 [ 0.85693837 -5.92788996  0.00825278 ...  0.19727835 -0.17868029
  -0.02871747]
 [-0.82308793  0.78071735 -1.15940766 ... -0.05662895 -0.06181358
  -0.12815981]]
```

利用 python 的 Sklearn 库的 LogisticsRegression 函数对训练集进行训练：

```
In [148]: from sklearn.linear_model import LogisticRegression
          classifier = LogisticRegression(random_state = 0, max_iter=10000)
          classifier.fit(X_train, y_train)

Out[148]: LogisticRegression(max_iter=10000, random_state=0)
```

对训练集进行预测：

```
In [151]: accuracy = (prediction[0][0] + prediction[1][1]) * 100.0 / prediction.sum()
          print("The final test accuracy is %.4f%%" % (accuracy))

          The final test accuracy is 99.8115%
```

对测试集进行预测：

```
In [152]: train_pred = classifier.predict(X_train)
          train_prediction = confusion_matrix(y_train,train_pred)
          print(train_prediction)
          train_accuracy = (train_prediction[0][0] + train_prediction[1][1]) * 100.0 / train_prediction.sum()
          print("The final accuracy on training data is %.4f%%" % (train_accuracy))

          [[60160    72]
           [  102 35178]]
          The final accuracy on training data is 99.8178%
```

训练后在训练集和预测集上都取得了 99.8%以上的准确率，说明所选特征以及训练方法比较准确。