《多元统计》期末报告

曾子轩

17344011

目录

1. 背景介绍 4
2. 数据概述 5
2.1 数据的读取 5
2.2 探索性分析 6
2.3 数据清洗 7
2.3.1 缺失值处理 7
2.3.2 分类变量处理 8
3. 数据分析 8
3.1 变量筛选 8
3.2 模型搭建 9
3.2.1 简单线性分类器 9
3.2.2 其它分类器 10
3.2.2.1 高斯核 SVM 10
3.2.2.2 决策树 11
3.2.2.3 随机森林 11
3.2.2.4 Adaboost 12
4. 结论 12
5. 后记 13
5.1 说明 13
5.2 最喜爱的模型 13
6. 附录 13

6.1 数据及项目来源.....13

6.2 代码.....13

1. 背景介绍

本报告研究的对象是旅客预订酒店的取消订单率，具体研究方法是通过给定的包括预定日期、预订房间数、过往预订历史等数据来预测特定旅客是否会取消自己的预订。

本报告数据来自 Kaggle 的 Hotel Booking Demand 项目（附录给出了项目网站地址）。下列数据字段的描述取自网站上该项目的介绍页：

Hotel：酒店类型

Is_canceled：该旅客是否取消，即本问题的标签变量

Lead_time：预订日期和入住日期的间隔

Arrival_date：包括四个子字段，表示的是入住日期

Stays_in_weekend_nights：预订居住的天数（周末）

Stays_in_week_nights：预订居住的天数（工作日）

Adults，Children，Babies：入住各类型旅客的人数

Meal：三餐预订情况

Country：旅客所属的国家或地区

Market_segment、Distribution_channel：预订来源，包括旅行社预订、私人预订等类型

Is_repeated_guest：是否有预订历史

Previous_cancellations、Previous_bookings_not_canceled：订单取消历史

Reserved_room_type：预订房间类型

Assigned_room_type：入住房间类型

Booking_changes：更改订单状态数

Deposit_type：押金类型

Agent：所属旅行社编号

Company：预订填写的单位

Days_in_waiting_list：预订排队天数

Custom_type：预订类型，包括合同预订和临时预订等

Adr：预订日均房价

Required_car_parking_spaces：是否有泊车需求

Total_of_special_requests：特殊要求数量

Reservation_status：预订状态

Reservation_status_date：预订状态最后变更日期

2. 数据概述

2.1 数据的读取

读取数据后，先查看数据各字段的类型，以及缺失值的情况。可以看到除了Company 一列，缺失值均在可接受范围内。

```
booking_changes                 119390 non-null int64
deposit_type                    119390 non-null object
agent                           103050 non-null float64
company                         6797 non-null float64
days_in_waiting_list            119390 non-null int64
customer_type                   119390 non-null object
adr                             119390 non-null float64
```

数据类型及缺失情况（部分）

再来查看标签值 Is_canceled 的分布情况（1 表示取消预订）。从下图可以看到标签变量并未出现严重的偏斜，表示可以正常抽样。

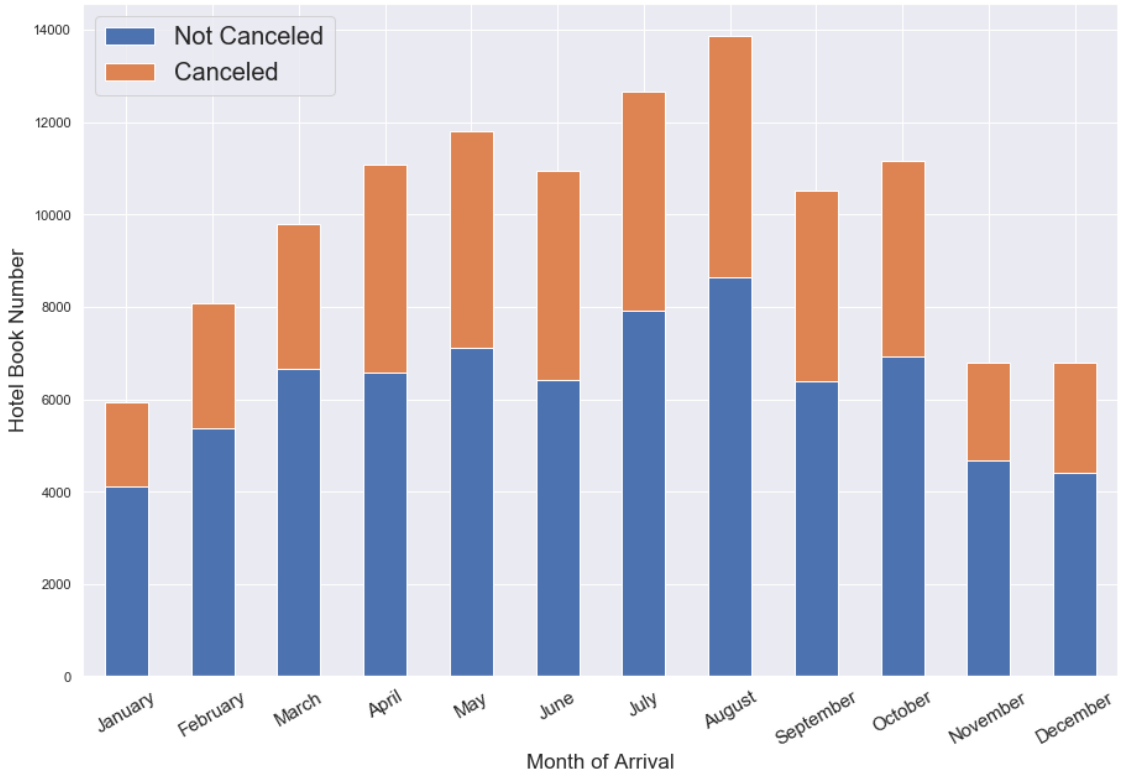
```
0    75166
1    44224
Name: is_canceled, dtype: int64
```

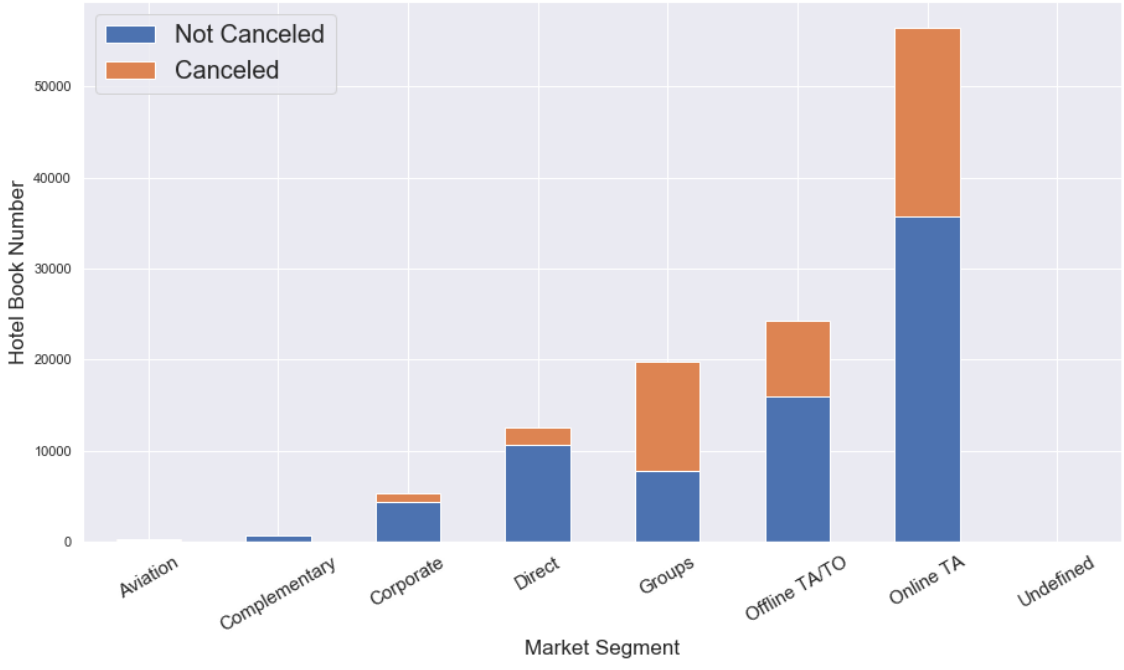
标签值的分布

2.2 探索性分析

在这里我们将对变量进行初步的分析，运用主观的判断或者条形图的绘制来筛选变量。由于变量众多，下面以几个典型变量为例。

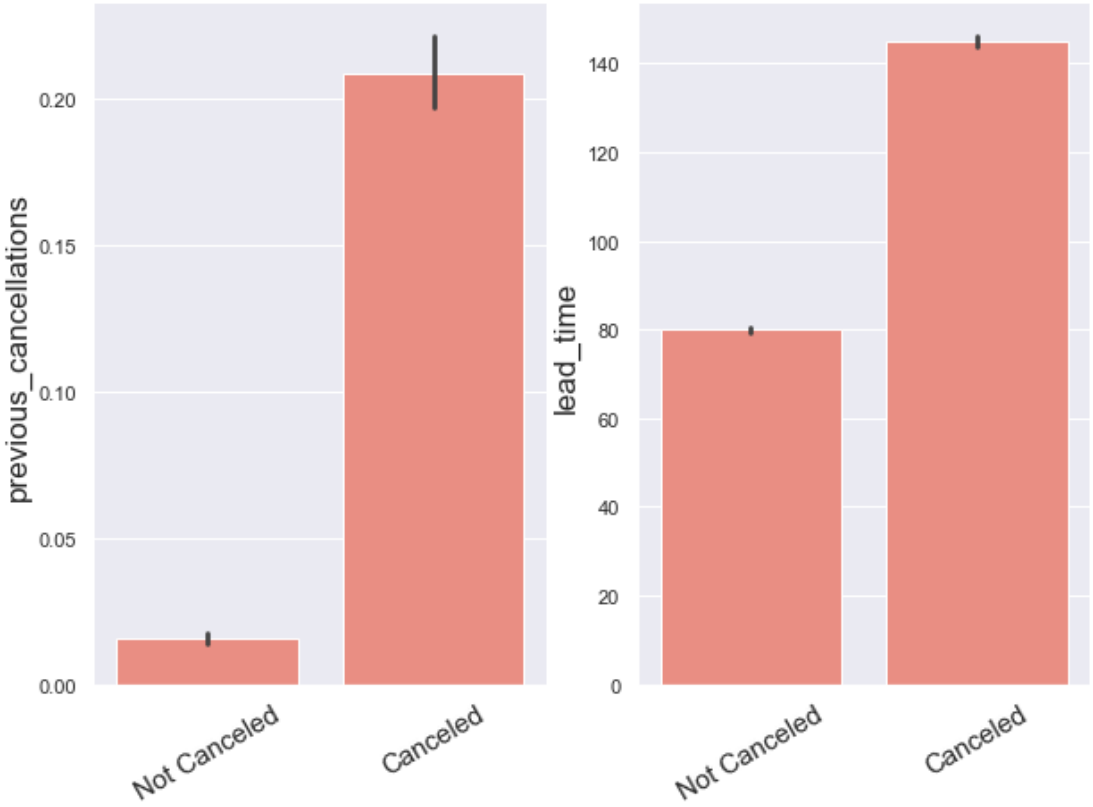
来看看旅客的入住月份和预订人类型（共同点：都是分类变量）对房间预订的取消率（以下简称取订率）的影响。





从以上二图可以看出，不同的入住月份和预订人的类型均对取订率有较大的影响；特别是不同类别的预订人，取订率可能差别巨大。因此应当考虑将以上两个变量均纳入考量指标。

再来看看一些连续变量的情况。



从上图可以看出，取消订单与否的两个人群，在取消订单历史和预订-入住时间间隔这两个变量水平上均存在较大差别，因此应考虑将这两个变量纳入考量指标中，这也符合我们的常识判断。

综合上面的办法，我们可以对变量进行粗略的分析和筛选，丢弃一些明显不相关的变量。

2.3 数据清洗

这部分主要是针对数据进行处理缺失值和分类变量的工作，为模型搭建做准备。

2.3.1 缺失值处理

对分类变量，考虑用众数填补缺失；对连续变量则考虑用中位数。

其中还需要注意一些细节：Agent和Company两列结合字段实际意义来看，缺失值应归纳为另一大类。如 Agent 的缺失值现实意义应为无“agent”，即私人预订，因此应该将缺失值视为一个新的类别。

2.3.2 分类变量处理

对分类变量，我们拟采用单热编码的方式将其数值化。在这里要注意一个分类变量如果有 c 个分类，那么单热编码的最终结果应该是生成 c-1 个列。

至此，我们完成了前期所有准备工作，得到了一个 119390 个样本，1017 个字段的变量矩阵，下面可以开始建模工作了。

3. 数据分析

3.1 变量筛选

首先，我们把数据集以 7：3 的比例切割为训练集和测试集，此处使用了以标签值为依据的分层切割。以下变量筛选方法使用的是训练集，测试集无影响。

现在，我们要最终确定将哪些变量加入模型。区别于 2.2.2 的直观分析和主观判断，现在我们将使用模型的办法来给出各变量与标签值的关联程度。

在随机森林框架（500 棵树，Bootstrap 每次取 119390 个样本和$\sqrt{1017}$个变量）下，我们对 is_canceled 一列进行预测，然后计算各变量所在的节点对于 Gini 值下降的贡献度，即 Gini-Importance，以此为依据得到了一个特征重要性指标。下面为该指标下，排名前五的变量：

	importance
lead_time	0.091770
deposit_type_Non Refund	0.078465
country_PRT	0.062140
adr	0.059912
total_of_special_requests	0.054008

最后我们决定选出排名前 10%的变量（100 个）加入模型。

3.2 模型搭建

3.2.1 简单线性分类器

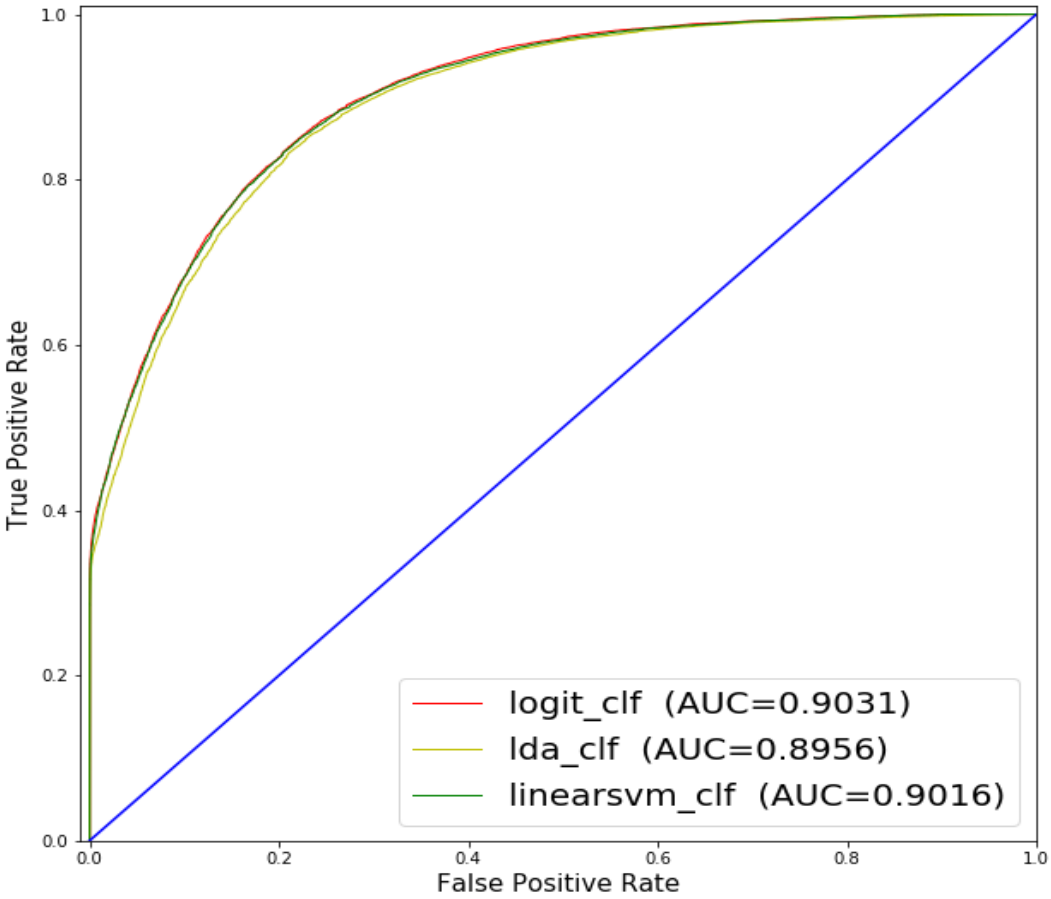
我们先使用几个简单的线性分类器（逻辑回归、LDA 和线性核 SVM）来进行分类，看看他们的效果。注意训练之前要先对数据做标准化处理。以下为各分类器的准确率一览：

```
 logit_clf 的准确率如下
训练集:  0.819367499072667
测试集:  0.820001675182176

 lda_clf 的准确率如下
训练集:  0.8079164323405885
测试集:  0.8081078817321383

 linearsvm_clf 的准确率如下
训练集:  0.818182905962452
测试集:  0.8186894491442611
```

绘制 ROC 曲线，并标出它们的 AUC 值：



显然，这些线性分类器的效果尚可，但我们仍想寻求更好的分类器。

3.2.2 其它分类器

我们来考虑使用其它类型的分类器。

3.2.2.1 高斯核 SVM

接下来使用高斯核 SVM 分类器训练模型。我们使用网格搜索技术来从一些可能的参数中（C 为 1,10,50,100,500,1000；gamma 为 10^-5,10^-4,10^-3,10^-2,10^-1）寻找一个最佳组合。由于原数据集过大导致的训练时间过长，为了示例方便起见，这里减少训练集的样本量至 5000。以下是五折检验得到的最佳参数以及对应的模型准确率：

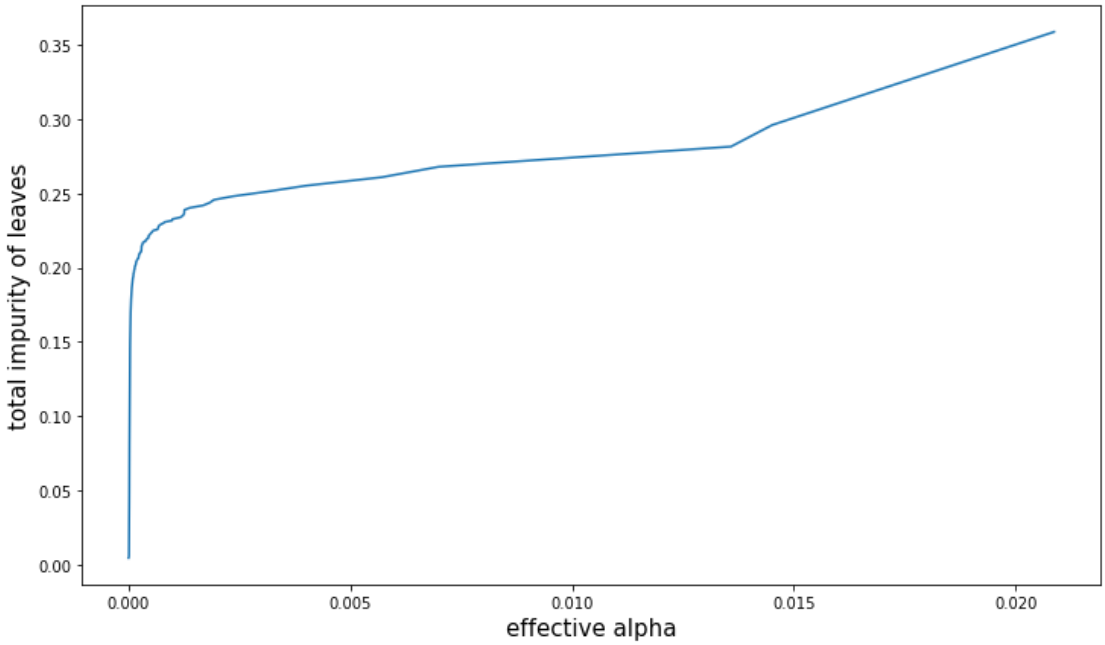
```
高斯核SVM最佳参数组合为： {'C': 10, 'gamma': 0.001}
最佳高斯核svm的准确率如下
训练集： 0.8504
测试集： 0.8187452885501298
```

可以看到高斯核的 svm 对比线性核并没有太大的起色。

3.2.2.2 决策树

下面转向一些非参数的分类方法，也就是树模型。

使用决策树分类器，计算得到各节点的 alpha 值，并绘图如下：



依据上图，我们选择 0.0001 作为 alpha 临界值来进行剪枝，剪枝后决策树表现如下：

```
tree_clf 的准确率如下
训练集:  0.8622162660189295
测试集:  0.8555155373146833
```

可以看到对比之前的模型，决策树在测试集上的表现有所提升，可见树模型是个不错的方向。

3.2.2.3 随机森林

现在来使用基于树分类器的集成学习方法。

首先使用随机森林进行分类工作。注意这次使用随机森林是为了直接输出预测结果，因此，区别于 3.1 我们理应增加一些限制来进一步提高泛化能力。在这

里增加的限制是森林中每棵树的深度最大为 30，其它参数与前述随机森林模型相同。准确率情况如下：

```
forest_clf 的准确率如下
训练集:  0.9736876742488603
测试集:  0.8879303124214758
OOB-SCORE:  0.8882414176827444
```

可以看到随机森林的拟合能力相比简单分类器强了许多，并且泛化能力更强。

3.2.2.4 Adaboost

再来看看提升树模型表现如何，这里选择经典的 Adaboost 方法。200 棵树，最大深度为 10，学习率为 0.5，以决策树为基分类器的 Adaboost 模型准确率如下：

```
ada_clf 的准确率如下
训练集:  0.9961111842341426
测试集:  0.8655387106681185
```

可见在此情景下 Adaboost 的拟合能力比随机森林更强（事实上由于更少的树和更多的正则化项，训练速度也要稍快），但泛化能力则略显不足。

4. 结论

至此，我们构建了多个分类器模型来预测旅客的取订与否，均取得了不错的效果。其中效果最好的是随机森林模型，在测试集上的准确率接近 89%。同时可以明显看出集成学习相对于简单的分类器，无论是在数据拟合能力还是模型泛化能力上，都具有较大的优势。

此外需要指出的是，本报告的模型有一些天生的缺陷。比如，由于数据量较大，很难使用网格搜索来得到最佳参数，因此各模型可能均没有达到最佳效果。另外，对一些变量的直接丢弃（而不是充分利用其中的信息）仍将不可避免地影

响所有模型的效果。

5. 后记

5.1 说明

1）本报告使用 Python 进行数据分析及建模的工作，代码见附录。

2）除了特别设定的参数外，各模型的参数均为 scikit-learn 机器学习库的默认参数。

5.2 最喜爱的模型

个人而言，我比较喜欢解释性强、对数据分布要求不高的非参数模型，因此我对随机森林、提升树这些模型是比较偏爱的。它不像一些线性分类器如逻辑回归、LDA，脱离了线性分界的场景，效果就会大幅度下滑；也不像深度网络一样有成百上千个参数，像一个不可捉摸的黑箱；另外树模型的训练速度比较快，可以大大减少调参时间。

为了使变量保持解释性，具有现实意义，我并没有采用 PCA 等无监督降维方法，而是使用了随机森林中的特征重要度这样一个指标来筛选变量。从几个树模型最终达成的不错的效果来看，自变量间的相关性并不大（因为决策树的决策边界都是正交的，所以用树模型选出来的变量相关性应该较小）。这也间接证明了【直接丢弃变量而不是寻求变量的重组降维】这一策略的合理性。

6. 附录

6.1 数据及项目来源

https://www.kaggle.com/jessemostipak/hotel-booking-demand

6.2 代码

注：代码运行环境是 python3 内核的 jupyter notebook

```
# Notebook 显示设置
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity='all'
import warnings
warnings.filterwarnings("ignore")
# 读取数据
import pandas as pd
file_path = r"C:\Users\Mac\Desktop\过程\临时文件\hotel_bookings.csv\hotel_bookings.
csv"
data = pd.read_csv(file_path, sep=',', header=0)
# 查看数据类型和缺失情况
data.info()
# 观察标签数据的分布情况
data['is_canceled'].value_counts()
# 备份数据
data_backup = data.copy()
# 绘制条形图 1
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
months = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'Septe
```

```
mber', 'October', 'November', 'December']
mapping = {month:i+1 for i, month in enumerate(months)}
data['month'] = data['arrival_date_month'].map(mapping)
data['count'] = 1
fig, ax = plt.subplots(1, figsize=[15, 10])
data.pivot_table(index='month', values='count', columns='is_canceled', aggfunc=np.s
um).sort_index().plot(kind='bar', stacked=True, ax=ax)
ax.set_xticklabels(months, fontsize=15, rotation=30)
ax.legend(["Not Canceled", "Canceled"], loc='upper left', fontsize=20)
ax.set_ylabel("Hotel Book Number", fontsize=17)
ax.set_xlabel("Month of Arrival", fontsize=17)
# 绘制条形图 2
fig, ax = plt.subplots(1, figsize=(15,8))
data.pivot_table(index='market_segment', values='count', columns='is_canceled', agg
func=np.sum).sort_index().plot(kind='bar', stacked=True, ax=ax)
ax.set_xticklabels(ax.get_xticklabels(), fontsize=15, rotation=30)
ax.legend(["Not Canceled", "Canceled"], loc='upper left', fontsize=20)
ax.set_ylabel("Hotel Book Number", fontsize=17)
ax.set_xlabel("Market Segment", fontsize=17)
# 绘制条形图 3
fig, axes = plt.subplots(1, 2, figsize=(10,7))
i = 0
```

```
for col in ['previous_cancellations', 'lead_time']:
    sns.barplot(x='is_canceled', y=col, data=data, ax=axes[i], color="salmon")
    axes[i].set_xticklabels(["Not Canceled", "Canceled"], fontsize=15, rotation=30)
    axes[i].set_xlabel("")
    axes[i].set_ylabel(col, fontsize=17)
    sns.despine()
    i += 1
# 恢复数据
data = data_backup
# 初步筛选掉一些明显不相关的变量
deleted_list=['arrival_date_year', 'reservation_status','reservation_status_date']
data.drop(columns=deleted_list, axis=1, inplace=True)
# 填补缺失值
data['children'].fillna(int(data['children'].mode()), inplace=True)
data['country'].fillna(str(data['country'].mode()), inplace=True)
data['agent'].fillna("none", inplace=True)
data['company'].fillna("none", inplace=True)
# 对分类变量进行单热编码
cat_list = ['hotel','arrival_date_month', 'arrival_date_week_number', 'arrival_date_day_
of_month',
                'meal', 'country', 'market_segment', 'distribution_channel', 'is_repeated_g
uest',
```

```
                'reserved_room_type','assigned_room_type', 'deposit_type', 'agent', 'com
pany',
                'customer_type', 'required_car_parking_spaces']
data = pd.get_dummies(data, columns=cat_list, prefix=dict(zip(cat_list, cat_list)), dro
p_first=True)
# 分割训练集和测试集
from sklearn.model_selection import StratifiedShuffleSplit
split = StratifiedShuffleSplit(n_splits=2, test_size=0.3, random_state=42)
for train_index, test_index in split.split(data, data['is_canceled']):
    train_data = data.iloc[train_index]
    test_data = data.iloc[test_index]
X_train = train_data.drop(columns=['is_canceled'], axis=1)
y_train = train_data['is_canceled']
X_test = test_data.drop(columns=['is_canceled'], axis=1)
y_test = test_data['is_canceled']
# 随机森林筛选变量
from sklearn.ensemble import RandomForestClassifier
clf_forest = RandomForestClassifier(n_estimators=500,n_jobs=-1)
clf_forest.fit(X_train, y_train)
importances = pd.DataFrame(clf_forest.feature_importances_, columns=["importance
"], index=X_train.columns).sort_values(by='importance', ascending=False)
X_train = X_train.loc[:, importances.index[:100]]
```

```
X_test = X_test.loc[:, importances.index[:100]]
# 标准化
from sklearn.preprocessing import StandardScaler
scale = StandardScaler()
scale.fit(X_train)
X_train = scale.transform(X_train)
X_test = scale.transform(X_test)
# 查看几个线性分类器的准确率
from sklearn.linear_model import LogisticRegression
logit_clf = LogisticRegression()
logit_clf.fit(X_train, y_train)
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
lda_clf = LinearDiscriminantAnalysis(solver='svd', n_components=2)
lda_clf.fit(X_train, y_train)
from sklearn.svm import LinearSVC
linearsvm_clf = LinearSVC()
linearsvm_clf.fit(X_train, y_train)
from sklearn.metrics import accuracy_score
def accuracy(clf_name, kind):
    clf = globals()[clf_name]
    if kind == 'train':
        y_pred = clf.predict(X_train)
```

```
            print("训练集：", accuracy_score(y_train, y_pred))
        if kind == 'test':
            y_pred = clf.predict(X_test)
            print("测试集：", accuracy_score(y_test, y_pred))
clf_list = ['logit_clf', 'lda_clf', 'linearsvm_clf']
for clf_name in clf_list:
    print("\n", clf_name, "的准确率如下")
    accuracy(clf_name, 'train')
    accuracy(clf_name, 'test')
# 绘制 ROC 曲线
from sklearn.metrics import roc_curve, roc_auc_score
import matplotlib.pyplot as plt
def plot_roc_curve(clf_name, linetype):
    clf = globals()[clf_name]
    try:
        scores = clf.decision_function(X_train)
    except:
        scores = clf.predict_proba(X_train)[:, 1]
    fpr, tpr, thresholds = roc_curve(y_train, scores)
    auc = roc_auc_score(y_train, scores)
    label = clf_name+"  (AUC=" + str(round(auc,4)) + ")"
    plt.plot(fpr, tpr, linetype, linewidth=1, label=label)
```

```
plt.figure(figsize=(10, 10))
for clf_name, linetype in zip(clf_list, ['r', 'y', 'g']):
    plot_roc_curve(clf_name, linetype)
plt.plot([0, 1], [0, 1], 'b')
plt.axis([-0.01, 1, 0, 1.01])
plt.xlabel('False Positive Rate', fontsize=15)
plt.ylabel('True Positive Rate', fontsize=15)
plt.legend(loc='bottomright', fontsize=20)
plt.show()
# 高斯核 svm（已减少数据量）
import random
index = random.sample(range(len(X_train)), 5000)
X = X_train[index, :]
y = y_train.iloc[index]
import numpy as np
from sklearn.svm import SVC
from sklearn.model_selection import KFold, GridSearchCV
fold = KFold(n_splits=5, random_state=42)
param_range = {'C': [1, 10, 50, 100, 500, 1000], 'gamma': np.logspace(-5, -1, 5)}
svm_clf = SVC(kernel='rbf')
grid= GridSearchCV(svm_clf, param_grid=param_range)
grid.fit(X, y)
```

```
print("高斯核 SVM 最佳参数组合为：", grid.best_params_)

print("最佳高斯核 svm 的准确率如下")

y_pred = grid.predict(X)

print("训练集：", accuracy_score(y, y_pred))

y_pred = grid.predict(X_test)

print("测试集：", accuracy_score(y_test, y_pred))

# 决策树剪枝图

from sklearn.tree import DecisionTreeClassifier

path = DecisionTreeClassifier(criterion='gini').cost_complexity_pruning_path(X_train, y

_train)

ccp_alphas, impurities = path.ccp_alphas, path.impurities

fig, ax = plt.subplots(1, figsize=(12, 7))

ax.plot(ccp_alphas[:-1], impurities[:-1])

ax.set_xlabel("effective alpha", fontsize=15)

ax.set_ylabel("total impurity of leaves", fontsize=15)

# 决策树

tree_clf = DecisionTreeClassifier(criterion='gini', ccp_alpha=0.0001)

tree_clf.fit(X_train, y_train)

print('tree_clf', "的准确率如下")

accuracy('tree_clf', 'train')

accuracy('tree_clf', 'test')

# 随机森林
```

```
forest_clf = RandomForestClassifier(n_estimators=500, max_depth=30, n_jobs=-1)

forest_clf.fit(X_train, y_train)

print('forest_clf', "的准确率如下")

accuracy('forest_clf', 'train')

accuracy('forest_clf', 'test')

print("OOB-SCORE：", forest_clf.oob_score_)

# Adaboost

from sklearn.ensemble import AdaBoostClassifier

ada_clf = AdaBoostClassifier(DecisionTreeClassifier(max_depth=10), n_estimators=20

0, learning_rate=0.5)

ada_clf.fit(X_train, y_train)

print('ada_clf', "的准确率如下")

accuracy('ada_clf', 'train')

accuracy('ada_clf', 'test')
```