【智慧化企業整合】

Project #2

怕不退，退不怕?

第 4 組

109034511 邱綉雅

109034542 林鈺婷

109034533 李珩慈

105062314 張峻騰

指導教授：邱銘傳 教授

目錄

一、背景介紹

1. 情境描述

面對激烈的競爭，飯店業者爭相推出彈性的預訂方案來吸引更多消費者入住，但這些便於旅客的政策下，卻隱藏著飯店業者因預訂取消而造成的空房危機，業者如何在彈性預訂政策和利益損失風險間取得平衡為本次我們欲探討的主題。

本專案預計透過 Kaggle 上的飯店預訂和需求公開資料，建立數據模型以嘗試預測較容易被取消的飯店預訂類型以及旅客因素，並以視覺化的圖表呈現預測結果使飯店業者可以更準確、快速地掌握飯店實際住房率。

2. 5W1H

What?	住房取消預測系統
When?	從顧客預訂到入住
Who?	旅宿業者
Where?	預訂網站、後台
Why?	住房取消無法及時重新安排，造成整體住房率降低
How?	資料分析、深度學習模型，預測住房是否取消

二、模型架構

1. 資料前處理過程

(1) 資料來源

由 Kaggle 的公開數據集中取得「飯店預訂資料集」作為模型的資料來源。資料集大小 (119210, 32)，共計 119210 筆住房資料，並且包含 32 個不同的資料欄位。(數據來源:https://www.kaggle.com/jessemostipak/hotel-booking-demand)

(2) 資料基本描述

我們依各個欄位的屬性將這些細項資料分為三大類別：顧客資料、訂單資料、住房資料。

資料類別	欄位名稱	資料型態	資料說明
顧客資料	Adult	數值	成人數量
	Babies	數值	嬰兒數量
	Children	數值	孩童數量
	Company	類別	公司型態
	CustomerType	類別	顧客型態
	Country	類別	國家型態
	IsRepeatedGuest	類別	預定是否為重複顧客的指標
	MarketSegment	類別	客戶的市場區隔

	PreviousBookingNotCanceled	數值	顧客之前未取消預定的數量
	PreviousCancellations	數值	顧客之前取消預定的數量
訂單資料	Agent	類別	代理商
	ADR	數值	每日平均房價
	ArrivalDateDayOfMonth	數值	到達日期之日期
	ArrivalDateMonth	類別	到達日期之月份
	ArrivalDateWeekNumber	數值	到達日期之星期數
	ArrrivalDateYear	數值	到達日期之年份
	BookingChanges	數值	預定的更改次數
	DaysInWaitingList	數值	等候訂單確定的時間
	DepositType	類別	存款類型
	DistributionChannel	類別	通路管道
	IsCanceled	類別	預定是否取消的指標
	LeadTime	數值	從預定到入住所經過之天數
	ReservationStatus	類別	預定狀態
	ReservationStatusDate	日期	預定狀態的更新日期
住房資料	AssignedRoomType	類別	預定之房型代碼
	Meal	類別	預定之餐點類型
	RequiredCarParkingSpaces	數值	顧客所需車位數量
	ReservedRoomType	類別	預定的房型
	StaysInWeekendNights	數值	入住或預定之週末過夜天數
	StaysInWeekNights	數值	入住或預定之平日過夜天數
	TotalOfSpecialRequestd	數值	顧客提出的特殊要求數量

(3) 空值處理

首先，我們使用 isnull()函式將所有空值的項目列出，並發現在 children, country, agent, company 欄位中有空值。因 children 的眾數為零，我們將空值以 0.0 填補，另外針對類別型的資料欄位 country, agent, company，我們用 Unknown 帶入，如下圖。

```
# check for missing values
full_data.isnull().sum()
```

```
hotel                               0
is_canceled                         0
lead_time                           0
arrival_date_year                   0
arrival_date_month                  0
arrival_date_week_number            0
arrival_date_day_of_month           0
stays_in_weekend_nights             0
stays_in_week_nights                0
adults                              0
children                            4
babies                              0
meal                                0
country                           488
market_segment                      0
distribution_channel                0
is_repeated_guest                   0
previous_cancellations              0
previous_bookings_not_canceled      0
reserved_room_type                  0
assigned_room_type                  0
booking_changes                     0
deposit_type                        0
agent                           16340
company                        112593
days_in_waiting_list                0
customer_type                       0
adr                                 0
required_car_parking_spaces         0
total_of_special_requests           0
reservation_status                  0
reservation_status_date             0
```

```
nan_replacements = {"children:": 0.0,"country": "Unknown", "agent": "Unknown", "company": "Unknown"}
full_data_cln = full_data.fillna(nan_replacements)
```

(4) 標準化

由於 adr 和 lead_time 兩資料欄位的值相對於其他資料欄位的值而言，規模過大，為避免影響模型準確率，因此採用資料標準化的方式，將值轉換為 zscore 的形式，統一數值規模，以利模型執行。

```
df['adr'] = zscore(df['adr'])
df['lead_time'] = zscore(df['lead_time'])
```

(5) 資料標籤化

在採用的資料欄位中，我們針對類別資料進行資料標籤化的處理。其中，因為 country 的資料有多種種類，而 hotel 的資料僅有兩種種類，因此使用了 label encoder 的方法將資料做轉換；而在 arrival_date_month 的資料欄位，則是因為希望資料標籤化的結果能夠從 1 開始，讓處理後的資料能夠更加直觀，因此選擇採用 map 的方法來做資料標籤化，剩餘其他的類別資料則皆採用 get_dummies 的方式來做資料標籤化的處理。

```
df = pd.get_dummies(df, columns = ["required_car_parking_spaces",
                                   "right_room",
                                   "far_in_advance",
                                   "recent_booking",
                                   "changed_booking",
                                   "previous_cancel",
                                   "special_requests",
                                   "arrival_date_year",
                                   "meal",
                                   "market_segment",
                                   "distribution_channel",
                                   "deposit_type",
                                   "customer_type",
                                   "reserved_room_type",
                                   "assigned_room_type"])

le = LabelEncoder()
df['country'] = le.fit_transform(df['country'])
df['hotel'] = le.fit_transform(df['hotel'])

df['arrival_date_month'] = df['arrival_date_month'].map({'January':1, 'February': 2,
                          'March':3, 'April':4, 'May':5, 'June':6, 'July':7,'August':8,
                          'September':9, 'October':10, 'November':11, 'December':12})
```

(6) 特徵值選取

藉由 pandas 內建的 corr()函數，我們觀察各欄位與”is_canceled”取消欄位的相關性，發現並沒有特徵值特別突出，因此我們僅將與”is_canceled”取消欄位連動”servation_status_date” ,”reservation_status”,”reserved_room_type”刪除。

```
cancel_corr = full_data.corr()["is_canceled"]
cancel_corr.abs().sort_values(ascending=False)[1:]
```

```
lead_time                        0.293123
total_of_special_requests        0.234658
required_car_parking_spaces      0.195498
booking_changes                  0.144381
previous_cancellations           0.110133
is_repeated_guest                0.084793
agent                            0.083114
adults                           0.060017
previous_bookings_not_canceled   0.057358
days_in_waiting_list             0.054186
adr                              0.047557
babies                           0.032491
stays_in_week_nights             0.024765
company                          0.020642
arrival_date_year                0.016660
arrival_date_week_number         0.008148
arrival_date_day_of_month        0.006130
children                         0.005048
stays_in_weekend_nights          0.001791
```

```
data_corr = data_corr.drop(['reservation_status','reservation_status_date'], axis=1)
```

2. 模型架構

(1) Machine Learning Model

為使模型的考慮更具全面性，首先以四種基本的 Machine Learning Model 來分析我們的資料，此四種模型分別為 DecisionTree、RandomForest、LogisticRegression、XGBoost，並以此作為參考點。

```
# define models to test:
base_models = [("DT_model", DecisionTreeClassifier(random_state=42)),
               ("RF_model", RandomForestClassifier(random_state=42,n_jobs=-1)),
               ("LR_model", LogisticRegression(random_state=42,n_jobs=-1)),
               ("XGB_model", XGBClassifier(random_state=42, n_jobs=-1))]
```

	DecisionTree	RandomForest	LogisticRegression	XGBoost
accuracy score	0.8246	0.8664	0.7937	0.8165

上表為四種模型的比較結果，由圖中可看出模型準確率最高的為 RandomForest，其準確率為 0.8664。

(2) Deep Learning Model

使用四個 Machine Learning Model 分析後，我們接著利用兩種 Deep Learning Model 做更進一步的模型訓練，此兩種模型分別是 Neural Network (NN)、Multi-layer Perceptron (MLP)。

■ **Neural Network + (NN+)**

NN+模型的基本架構為一輸入層(relu)、一隱藏層(relu)及一輸出層(sigmoid)，我們自行再加入了一層 dropout(alpha = 0.25)以忽略部分資料。

所給定之基本參數為:

- 第一層為 500 個節點，第二層為 100 個節點
- gradient descent optimizer ＝ 'adam'
- loss function ＝ 'binary_crossentropy'
- 進行一次 50 個 epochs 的訓練

```
#NN+
from keras.layers import Dropout
from keras.models import Sequential
from keras.layers import Dense
model=Sequential()
model.add(Dense(500,input_dim=11,activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(100,activation='relu'))
model.add(Dense(1,activation='sigmoid'))
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
model.fit(X_train,y_train,epochs=50,batch_size=35,validation_data=(X_test,y_test))
```

```
2612/2612 [==============================] - 5s 2ms/step - loss: 0.4224 - accuracy: 0.8041
1120/1120 [==============================] - 2s 2ms/step - loss: 0.4282 - accuracy: 0.8011
Train accuracy:0.804
Test accuracy:0.801
```

■ Multi-layer Perceptron (MLP)

MLP 為一種前向傳遞類神經網路，利用倒傳遞的技術達到監督式學習包含了三層結構：輸入層、隱藏層、輸出層，為 DNN 的一種特例。我們利用 sklearn 套件中的 MLP 分類器，調整當中的各個參數來訓練模型。

所給定之基本參數為:

- hidden_layer_sizes＝（X.shape[1]）
- activation＝ ‘relu’
- solver＝ ‘adam’
- batch_size＝ ‘auto’
- learning_rate＝ ‘constant’
- learning_rate_init＝0.001
- max_iter＝200
- random_state＝0

```
from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier(hidden_layer_sizes=(X.shape[1]),
activation='relu',
solver='adam',batch_size='auto',
learning_rate='constant',
learning_rate_init=0.001,
max_iter=200,
random_state=0)

#  mlp.summary()
```

```
mlp.fit(X_train,y_train)
print("Train accuracy of MLP:{:.3f}".format(mlp.score(X_train,y_train)))
print("Test accuracy of MLP:{:.3f}".format(mlp.score(X_test,y_test)))
```

```
Train accuracy of MLP:0.824
Test accuracy of MLP:0.819
```

```
from sklearn.metrics import f1_score
predict = mlp.predict(X_test)
print(f1_score(predict, y_test))
```

```
0.7230880908663905
```

3. 模型小結

根據前述所列的模型進行訓練後，因四種 Machine Learning Model 中，RandomForest 準確度最高，因此我們將此模型拿來和 Deep Learning Model 做比較，所得到的結果如下表所示：

	RandomForest	NN+	MLP
學習程度	Machine Learning	Deep Learning	Deep Learning
套件	sklearn	keras	sklearn
調整空間	少	多	多
調整速度	快	慢	中
準確度	86.64%	80.1%	81.9%

由表可發現 RandomForest 的準確度較 NN+和 MLP 模型高，因此接下來我們將做更進一步的資料優化及資料重分配，以改善模型的預測準確率。

三、資料優化

在前期調整模型參數的部分，在經過各種參數調整後，無論是 NN+模型，或是 MLP 模型，在準確率的部分皆無法再得到更高的改善，且在 NN+模型中，每次 iteration 的準確率中多有高度的變異，為求更高的準確率，我們針對資料做了一些處理。 首先，我們先找出對於訂房是否取消影響較大的資料欄位，保留影響較大前九項資料欄位，再做參數調整，雖然在每一個 iteration 中能夠得到較穩定的準確率，但是在最終準確率的部分卻沒有提升。因此，我們依資料的型態和分別對不同欄位進行離群值處理及資料重分類。

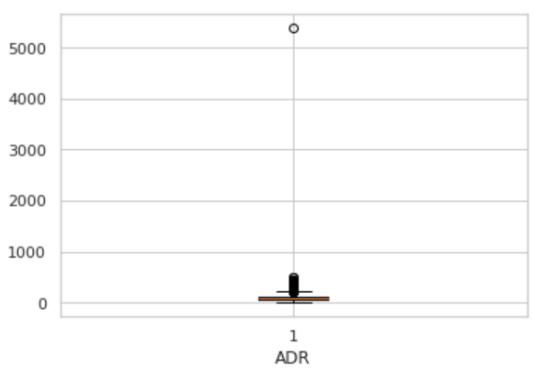
1. 離群值處理

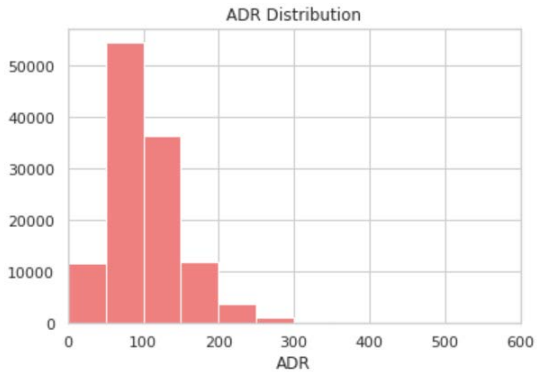
我們重新檢視資料，發現資料中存在些許離群值，因此我們嘗試去除離群值資料，而在去掉這些離群值資料後，仍保有約 95%的資料，資料量充足，再利用這些資料重新測試模型，得到了較高且穩定的準確率。我們使用 pandas.decribe()函數以及觀察圖表如盒鬚圖、直方圖、長條圖後發現在資料中，有些欄位的離群值相當明顯，包括 ADR、Distribution Channel、Room Type、Deposit Type，以下我們將針對這些特徵值說明其資料分布和處理方式。

	is_canceled	lead_time	arrival_date_year	arrival_date_week_number	arrival_date_day_of_month
count	119390.000000	119390.000000	119390.000000	119390.000000	119390.000000
mean	0.370416	104.011416	2016.156554	27.165173	15.798241
std	0.482918	106.863097	0.707476	13.605138	8.780829
min	0.000000	0.000000	2015.000000	1.000000	1.000000
25%	0.000000	18.000000	2016.000000	16.000000	8.000000
50%	0.000000	69.000000	2016.000000	28.000000	16.000000
75%	1.000000	160.000000	2017.000000	38.000000	23.000000
max	1.000000	737.000000	2017.000000	53.000000	31.000000

■ **ADR**

在 ADR 的部分，我們發現資料值大多落在 300 內，且因 ADR 為每日平均房價，正常情況下不應為負值。因此我們將低於 0 和高於 300 的資料刪除。

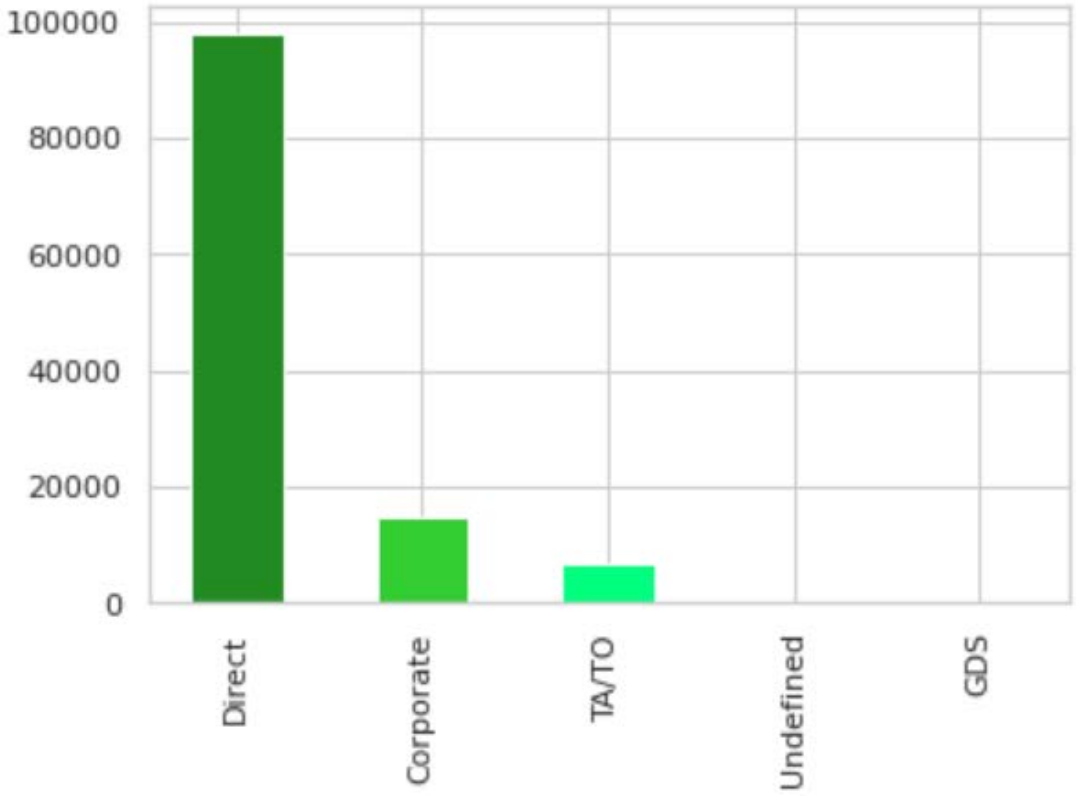





```
data = data[data.adr >=0 ]
data = data[data.adr <= 300]
```

■ **Distribution Channel**

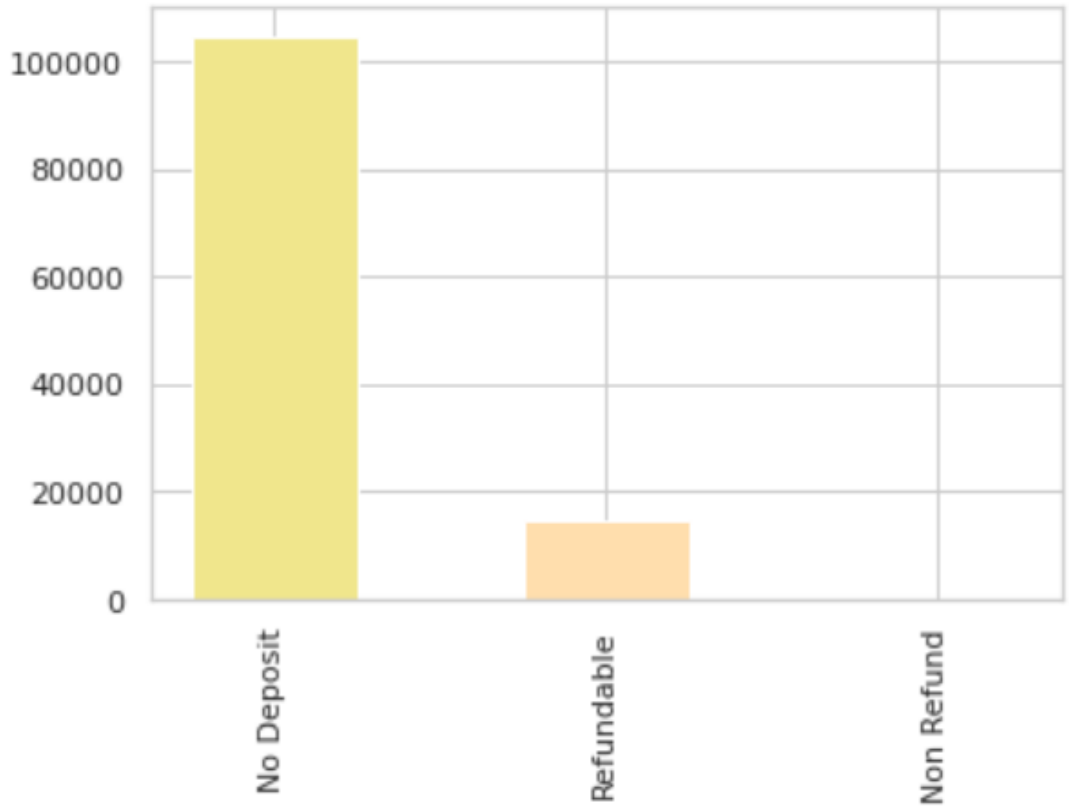
在 Distribution Channel 的部分，我們發現只有少部分資料為 Undefined 和 GDS，因此我們將這兩類別的資料刪除。




```
data = data[(data['distribution_channel'] == 'TA/TO') |
            (data['distribution_channel'] == 'Direct') |
            (data['distribution_channel'] == 'Corporate')]
```

■ **Deposit Type**

在 Deposit Type 的部分，我們發現 Non Refund 類別的資料佔絕少數，因此我們將此類別的資料刪除。




```
data = data[data.deposit_type != 'Non Refund']
```

2. 資料重分類

我們發現 Booking Changes、Parking Space、Special Request、Previous Cancel 四個類別的資料皆為 0 居多，並從 1 逐步下降資料頻率，因此我們認為可以將這四個類別轉換為 binary 型態，分別訂為有無訂單更改、有無車位需求、有無特殊需求、有無過去取消紀錄來解釋。另外，在 Lead time 類別，我們以 25 百分位數前和 75 百分位數後的資料分別將其定義為最近預定和提早預訂兩個 binary 類別。

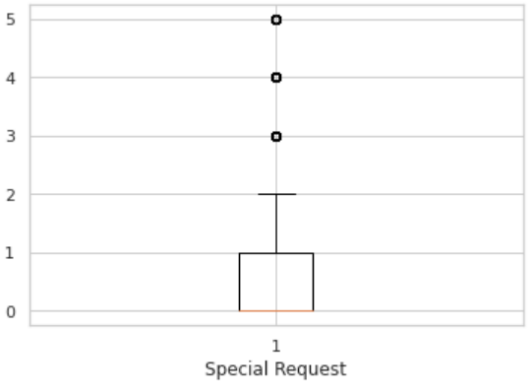
■ Booking Changes

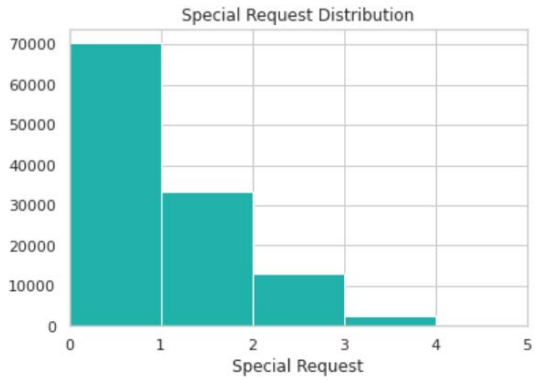

```
data['changed_booking']  =  pd.Series([0  if  x  ==  0  else  1  for  x  in  data['booking_changes']])
data['changed_booking']  =  pd.Categorical(data['changed_booking'])
```

■ Parking Space


```
data['required_car_parking_spaces']  =  pd.Series([0  if  x  ==  0  else  1  for  x  in  data['required_car_parking_spaces']])
data['required_car_parking_spaces']  =  pd.Categorical(data['required_car_parking_spaces'])
```

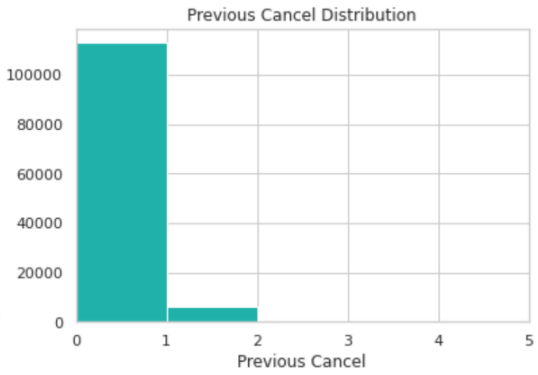
■ Special Request






```
data['special_requests']  =  pd.Series([0  if  x  ==  0  else  1  for  x  in  data['total_of_special_requests']])
data['special_requests']  =  pd.Categorical(data['special_requests'])
```

■ Previous Cancel




```
data['previous_cancel'] = pd.Series([0 if x == 0 else 1 for x in data['previous_cancellations']])
data['previous_cancel'] = pd.Categorical(data['previous_cancel'])
```

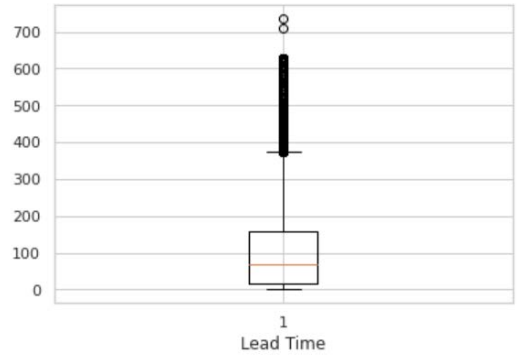
■ Lead time

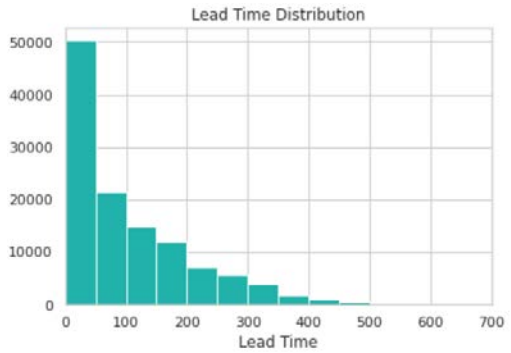
```
#lead time distribution

bins_list=[-50, 0, 50, 100, 150, 200, 250, 300, 350, 400, 450, 500, 550, 600, 650, 700]
n, bins, patches=plt.hist(full_data_cln.lead_time, bins=bins_list, color = 'lightcoral')
print(n)
print(bins)
for p in patches:
        print(p)
plt.title('Lead Time Distribution')
plt.xlim((0, 700))
plt.xlabel('Lead Time')
plt.show()
```

```
#lead time boxplot

plt.boxplot(full_data_cln.lead_time)
plt.xlabel('Lead Time')
plt.show()
```






```
data['far_in_advance'] = pd.Categorical(np.where(data['lead_time'] >= 180, 1, 0))
data['recent_booking'] = pd.Categorical(np.where(data['lead_time'] <= 16, 1, 0))
```

四、模型分析

1. Neural Network +(NN+)

dense	500,relu
dropout	0.25
dense	100,tanh
dropout	0.2
dense	1,softmax
loss_function	mean_squared_error
optimizer	sgd
epochs	constant
batch_size	constant
train_accuracy	0.63
test_accuracy	0.628

dense	500,relu
dropout	0.25
dense	300,tanh
dropout	0.2
dense	1,softmax
loss_function	mean_squared_error
optimizer	sgd
epochs	constant
batch_size	constant
train_accuracy	0.63
test_accuracy	0.628

dense	500,relu
dropout	0.25
dense	100,relu
dropout	0.2
dense	1,softmax
loss_function	mean_squared_error
optimizer	sgd
epochs	constant
batch_size	constant
train_accuracy	0.63
test_accuracy	0.628

dense	500,relu
dropout	0.25
dense	100,relu
dropout	0.2
dense	1,sigmoid
loss_function	mean_squared_error
optimizer	sgd
epochs	constant
batch_size	constant
train_accuracy	0.63
test_accuracy	0.628

dense	500,relu
dropout	0.25
dense	100,relu
dropout	0.2
dense	1,sigmoid
loss_function	binary_crossentropy
optimizer	sgd
epochs	constant
batch_size	constant
train_accuracy	0.63
test_accuracy	0.628

dense	500,relu
dropout	0.25
dense	100,relu
dropout	0.2
dense	1,sigmoid
loss_function	binary_crossentropy
optimizer	adam
epochs	constant
batch_size	constant
train_accuracy	0.869
test_accuracy	0.862

在 NN+模型中，我們調整參數，嘗試不同的參數組合對於模型準確率的影響，目標為最大化模型準確率。

從上圖可以看出，一開始更改第一層 dense、第二層 dense 以及 loss_function 的參數時，模型準確率皆維持在 0.628，並沒有變動。再嘗試多組參數後，我們決定將 optimizer 從 sgd 調整為 adam，最終模型準確率提升至 0.862。

```
#NN+
from keras.layers import Dropout
from keras.models import Sequential
from keras.layers import Dense
model=Sequential()
model.add(Dense(500,input_dim=X.shape[1],activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(100,activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(1,activation='sigmoid'))
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])

model.fit(X_train,y_train,epochs=50,batch_size=35,validation_data=(X_test,y_test))
```

```
2480/2480 [==============================] - 3s 1ms/step - loss: 0.2808 - accuracy: 0.8690
1063/1063 [==============================] - 1s 1ms/step - loss: 0.3031 - accuracy: 0.8618
nTrain accuracy:0.869
nTest accuracy:0.862
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_6 (Dense)              (None, 500)               32500
_________________________________________________________________
dropout_4 (Dropout)          (None, 500)               0
_________________________________________________________________
dense_7 (Dense)              (None, 100)               50100
_________________________________________________________________
dropout_5 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_8 (Dense)              (None, 1)                 101
=================================================================
Total params: 82,701
Trainable params: 82,701
Non-trainable params: 0
_________________________________________________________________
```

2. Multi-layer Perceptron (MLP)

hidden_layer_size	X.shape[1]
activation	relu
solver	adam
batch_size	auto
learning_rate	constant
learning_rate_init	0.001
max_iter	200
train_accuracy	0.875
test_accuracy	0.865
F1_score	0.8098

hidden_layer_size	X.shape[1]
activation	relu
solver	sgd
batch_size	auto
learning_rate	constant
learning_rate_init	0.001
max_iter	200
train_accuracy	0.836
test_accuracy	0.836
F1_score	0.7663

hidden_layer_size	X.shape[1]
activation	tanh
solver	adam
batch_size	auto
learning_rate	constant
learning_rate_init	0.001
max_iter	200
train_accuracy	0.879
test_accuracy	0.862
F1_score	0.8164

hidden_layer_size	X.shape[1]
activation	logistic
solver	adam
batch_size	auto
learning_rate	constant
learning_rate_init	0.001
max_iter	200
train_accuracy	0.885
test_accuracy	0.868
F1_score	0.821

hidden_layer_size	X.shape[1]
activation	logistic
solver	adam
batch_size	auto
learning_rate	constant
learning_rate_init	0.001
max_iter	1000
train_accuracy	0.888
test_accuracy	0.87
F1_score	0.8219

hidden_layer_size	X.shape[1],40
activation	logistic
solver	adam
batch_size	auto
learning_rate	constant
learning_rate_init	0.001
max_iter	1000
train_accuracy	0.892
test_accuracy	0.873
F1_score	0.8233

在 MLP 模型中，我們調整 sklearn 套件中的 MLP 分類器的參數，嘗試不同的參數組合對於模型準確率的影響，目標為最大化模型準確率。

首先，我們嘗試將 solver 從 adam 調整為 sgd，其準確率從 0.865 下降至 0.836，因此我們將 solver 改回 adam 並將 activation 從 relu 改為 tanh，使準確率從 0.836 上升至 0.862，接著再嘗試將 activation 從 tanh 改為 logistic，此改變讓準確率提升至 0.868，接著我們設定 max_iter 為 1000，準確率提升至 0.87，最後將 hidden_layer_sizes 改為兩層，使準確率最終提升到 0.873。

```
from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier(hidden_layer_sizes=(X.shape[1],40),
activation='logistic',
solver='adam',batch_size='auto',
learning_rate='constant',
learning_rate_init=0.001,
max_iter=1000,
random_state=0)

```

```
mlp.fit(X_train,y_train)
print("Train accuracy of MLP:{:.3f}".format(mlp.score(X_train,y_train)))
print("Test accuracy of MLP:{:.3f}".format(mlp.score(X_test,y_test)))
```

```
Train accuracy of MLP:0.892
Test accuracy of MLP:0.873
```

```
from sklearn.metrics import f1_score
predict = mlp.predict(X_test)
print(f1_score(predict, y_test))
```

```
0.8233314177817914
```

3. 模型比較

經過資料優化後，我們分別調整兩種 Deep Learning Model 的參數，並且不斷嘗試各種參數組合，期望能使準確率有所提升。下表為模型在不同階段的準確率以及進步幅度：

模型	Training Process	資料前處理	資料優化	Optimal Model	Improvement
NN+	• activation • optimizer • loss_function • epochs • batch_size	0.801	0.862	0.862	+6.1%
MLP	• activation • solver • max_iter • hidden_layer_sizes	0.819	0.865	0.873	+5.4%

五、結果與討論

1. 結論

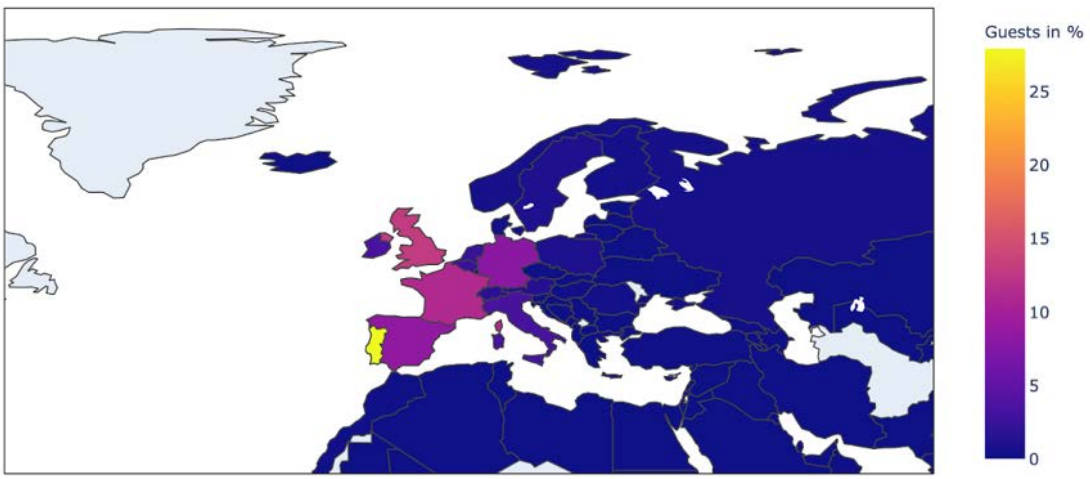
現今飯店業者時常因預訂取消而造成空房危機，因此本專案採用 Kaggle 上的飯店預訂和需求公開資料，透過資料前處理、模型分析、套用機器學習與深度學習模型等，來預測飯店的預訂取消。第一階段預測時，我們發現機器學習模型比深度學習模型的預測準確率更高，因此我們決定做更進一步的資料優化，期望能更準確的預測出飯店預訂的取消。

再資料優化過後，經由不斷嘗試來調整適合的參數，並觀察每次調整過後的結果，從中找出一組最適當的參數來訓練模型，由結果可以看出，兩種深度學習模型的預測結果皆有顯著的提升，比原本更加準確。

2. 未來展望

本專案針對飯店的預定取消進行模型訓練及預測，由結果可看到，雖然兩種深度學習模型的最佳預測結果有 87.3%，但仍然沒有到非常準確。我們認為，仍然有許多外部因素會影響顧客的飯店預定取消，例如：天氣狀況、重大事件等等。因此未來應該將多種外部因素考慮進模型內，進行更複雜的模型訓練，如此應能提升飯店預定取消的預測準確度。

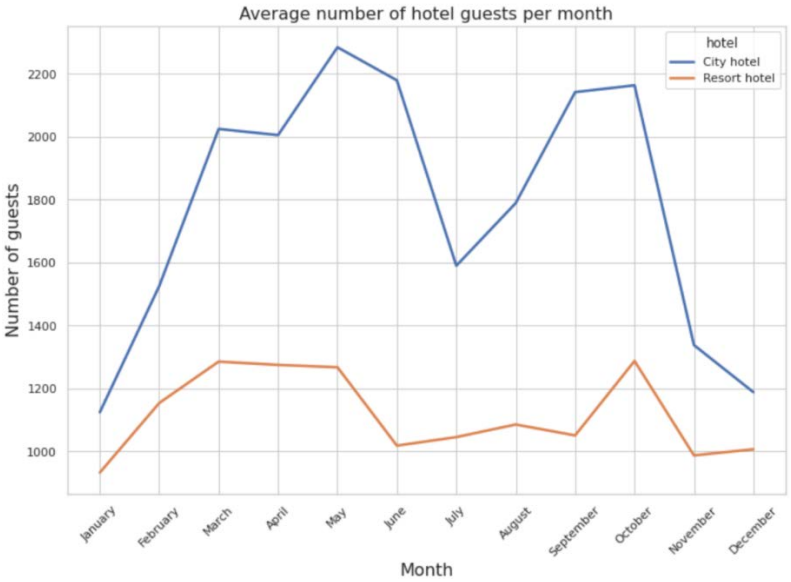
■ 住房地區分析



由於本次的資料集是來自葡萄牙的飯店業者，我們可以看到顧客多來自於歐洲各國家，然而因為歐洲屬於疫情的重災區，飯店的住房取消率可能會跟著提升，因此我們的預測模型也應該加入疫情因素，使飯店業者在預測預訂取消率上有更高的準確率。

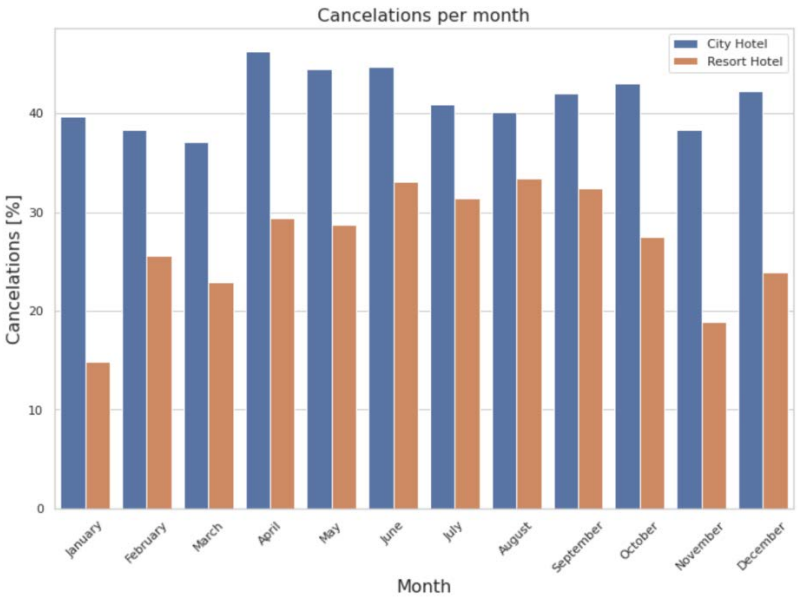
若未來能將我們訓練出的模型實際套用在各大飯店，將能使業者及早得知顧客是否有可能取消訂房，並做出相對應的措施。以下針對幾張資料視覺化後的圖表說明可發展的相對應措施：

■ **平均每月住客數**



由上圖可以看出，由資料整理出的每月住客數有明顯的季節性，每年的 5 月和 9 月為住房的高峰期，此時飯店業者可超賣房間做為因應的措施。而飯店業者可以根據我們的模型來預測住房預訂取消率，並根據預測結果來決定超賣房間的數量、提早準備備品等，藉此降低空房所造成的收益損失。

■ **每月住房取消比率**



由上圖可以看出，City Hotel 的預訂取消率比 Resort Hotel 高，取消率較高的話，所需的人力就會下降，因此人力資源配置可能需有所調整。飯店業者可以根利用我們的模型來預測預定取消比率，並根據取消率的高低來彈性調整人力資源，以節省不必要的人力成本支出。