數據科學與大數據分析
期末報告

第一組

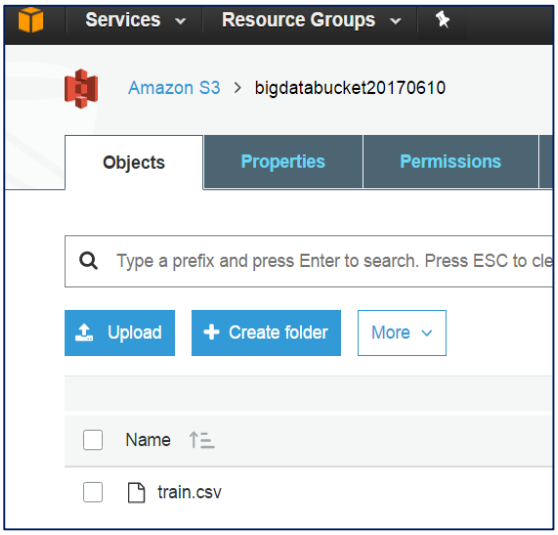
資科三　蔡漢龍 103703019 (雲端建置)
金融碩一　沈柏宇 105352034 (點擊率預測)
統計碩二　梁家安 104354031 (干擾因子、營收預測)

Github：https://github.com/BigDataAnalytics2017/ProjectOTA

目錄

一、 雲端建置 — AWS & GCP

使用 **AWS S3 服務存放 train.csv**，利用 **python 的 boto3 package** 直接讀檔，如此一來，在不同的雲端虛擬機環境下都能抓到資料，不必再每次新開虛擬機時重新上傳檔案。




```
import boto3
bucket = "bigdatabucket20170610"
file_name = "train.csv"
s3 = boto3.client('s3')
obj = s3.get_object(Bucket= bucket, Key=file_name)
train = pandas.read_csv(obj['Body'])
```

環境使用 **GCP** 提供的 **Dataproc 服務**去建置 cluster，一開始規格選擇 4vCPU 15GB memory 的 Master 搭配兩個 4vCPU 7.5GB memory 的 slave，觀察後發現記憶體不足，最後是用 **8vCPU 30GC memory** 的 Master 與兩個 slave 成功跑完 2.2GB 的 train.csv。

1.1 Apache Spark 2.0.2
Apache Hadoop 2.7.3
Apache Pig 0.16.0
Apache Hive 2.1.1
GCS connector 1.6.1-hadoop2
BigQuery connector 0.10.2-hadoop2

cluster-2

Overview Jobs VM Instances Configuration

Edit

Name	cluster-2
Zone	asia-southeast1-a
Master node	Standard (1 master, N workers)
Machine type	n1-standard-8 (8 vCPU, 30.0 GB memory)
Primary disk size	500 GB
Worker nodes	2
Machine type	n1-standard-8 (8 vCPU, 30.0 GB memory)
Primary disk size	500 GB
Local SSDs	0
Preemptible worker nodes	0
Cloud Storage staging bucket	dataproc-dbfb4fbb-89d0-44a2-801b-5c3bd70f0b00-asia-southeast1
Network	default
Image version	1.1.32
Project access	Allow API access to all Google Cloud services in the same project
Created	Jun 17, 2017, 9:04:10 PM

Equivalent REST

在 Dataproc 中有直接提供各個版本的映像檔讓使用者選擇，我們選用的是 **1.1 版**，內含的套件如上圖所示，確認後大概十分鐘便可建置完成，非常方便，再安裝 **Anaconda** 去管理 python 的 package，並把需要的都安裝好，便可使用 **spark-submit** 去執行程式。

過程中 AWS 與 GCP 的雲端虛擬機都有成功建立，最後之所以**選用 GCP**，有以下幾點理由：

- **GCP 的網頁比 AWS 還人性化**：以圖形替代繁複的文字描述，操作起來不會像 AWS 常常找不到目標。
- **Cluster 建立過程簡單且速度快**：如上面所提到的，GCP 只需要選好規格與映像檔版本，按下確認便可以成功建立；AWS 建置時要先產生 user 與 group，設定存取規則，確認虛擬機地區(ex. us-east-a)與所要用的 key 一致，之後建置指令所有參數都要一一確認網站上的說明書，spark 等套件還要自己下載放進去，完全弄好大概要花一個小時，對還在測試記憶體需求所以不斷建置的我們來說，太花時間。
- **虛擬機開啟容易**：GCP 只需點擊滑鼠左鍵，便可利用 Google chrome 開啟，對電腦毫無負擔；而 AWS 要開 ubuntu 連接，還要確認 key 與 pem，步驟繁瑣。
- **沒有 Access Key**：使用 Access Key 雖然有讓組員能一同使用而不需分享真實帳密的好處，但也會產生安全上的顧慮，深有體會。這次在建置 AWS 虛擬機的過程中，一邊將所有指令紀錄於記事本，包含 Access Key 的內容，結果一不小心就把它同步到了 github 上，僅僅十分鐘就遭到盜用，還好立即發現因此沒有損失，可見得現在網路爬蟲的可怕。

二、 點擊率預測 — Python 2.7 scikit-learn

我們的目標是**預測點擊率(click_bool)**，由於 Expedia 的 test.csv 資料，沒有 click_bool、booking_bool 的真實數值，無法檢測模型的成效，因此我們的 **Training data、Testing data 都是從 train.csv 的資料做切割**，特徵值的選擇排除 click_bool、booking_bool、gross_booking_usd (使用 click_bool 或 gross_booking_usd 預測 booking_bool 都是不合理的)，程式涵蓋 input data → data preprocessing → model training (including CV) & testing → output result，一貫的自動化流程，因此在雲端平台的運算需要比較多資源及時間(共一個半小時左右)。

■ 特徵值處理與選取

根據 Expedia 對特徵值的說明，我們主觀的處理特徵值、填補 null(處理後仍有 null 的資料則剔除)，挑選的特徵值在附圖(2.1) Feature Importance 一併呈現。

(1) date_time

→ 分拆成 date_year、date_month、date_day、date_week 四個特徵值

(2) visitor_hist_starrating：該消費者對過去所有消費過的飯店，所給予的平均星等；null 代表未曾消費。

→ null 設為 0 (仿效 prop_starrating)

(3) visitor_hist_adr_usd：過去該消費者平均每晚消費多少 USD；null 代表未曾消費。

→ null 設為 0

(4) prop_location_score2：飯店位置的優劣勢

→ null 設為 0，再與 prop_location_score1 加總

(5) srch_query_affinity_score：在搜尋引擎上被點擊的機率 (log 值)

→ 轉成機率，null 設為 0

(6) orig_destination_distance：在搜尋的當下，消費者與飯店的地理距離；null 代表無法計算。

→ 屬於 **semi-supervised learning**，用 site_id、visitor_location_country_id、prop_country_id、srch_destination_id 做 **Clustering (K-means)**，計算個別群集的平均值填補 null

(7) rate_percent_diff：將 8 間競爭者的 comp_rate、comp_rate_percent_diff 合併為一個特徵值。

→ $rate\_percent\_diff = \sum_{i=1}^{8} comp\ i\ \_rate \times comp\ i\ \_rate\_percent\_diff$

(8) comp1_inv (其餘 comp2~8 處理方式相同)：所搜尋的飯店是否為 Expedia 所特有，在競爭者 comp1 的網站沒有。

→ null 設為 0

■ **資料處理與切割**

由於 Click 資料占比非常小 (<10%)，為了解決資料的不對稱性，資料切割的方式主要分為兩階段：

(1) 所有資料分為 Click、not Click 兩類，個別抽出 **60%**的資料量合併為 Training data set，剩餘的 **40%**作為 Testing data set。

(2) Training data set 裡面的 **Click 樣本**再做**重複抽樣**，使得 Click、not Click 的資料量相等，作為最後的 Training data set，後續做 cross validation。

■ **模型**

我們用 Decision tree 的 Ada-Boost 模型(**AdaBoostClassifier**)預測 **Click_bool**，使用 **5-fold cross validation**，挑選出來的 parameters 與程式默認的一樣 (maximum number of trees = 50；shrinkage parameter = 1.)。

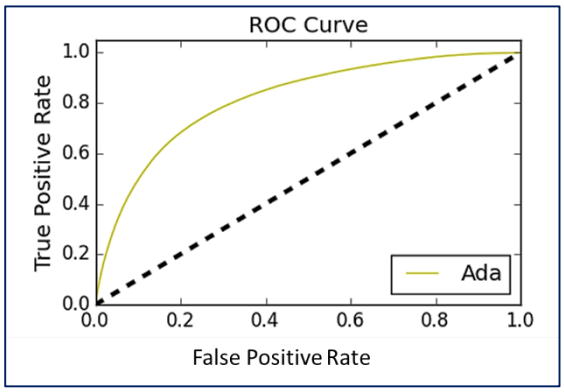
■ **結果與探討**

從附圖(2.1) Feature Importance 可以看到，比較重要的特徵值包括 **position**(飯店的地點)、**price_usd**(瀏覽時的報價)，而我們彙整的 **rate_percent_diff**(競爭者之間的價格競爭)也有一定的解釋力，但 **date_year**、**date_month**、**date_day**、**date_week** 的表現卻很差，我們認為將這三者變數轉為 Dummy variable，才能體現他們的解釋力，單純的數值型態會淪落為 binomial feature，因此我們額外拿部分資料做測試，結果請見附圖(2.2)。

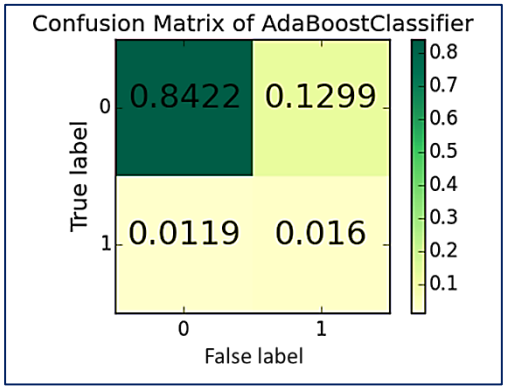
AdaBoostClassifier report

	precision	recall	f1-score	support
not Click	0.9861	0.8664	0.9224	3850574
Click	0.1098	0.5741	0.1844	110586
avg / total	0.9616	0.8582	0.9017	3961160

Time: 0 days 01:14:26.243478

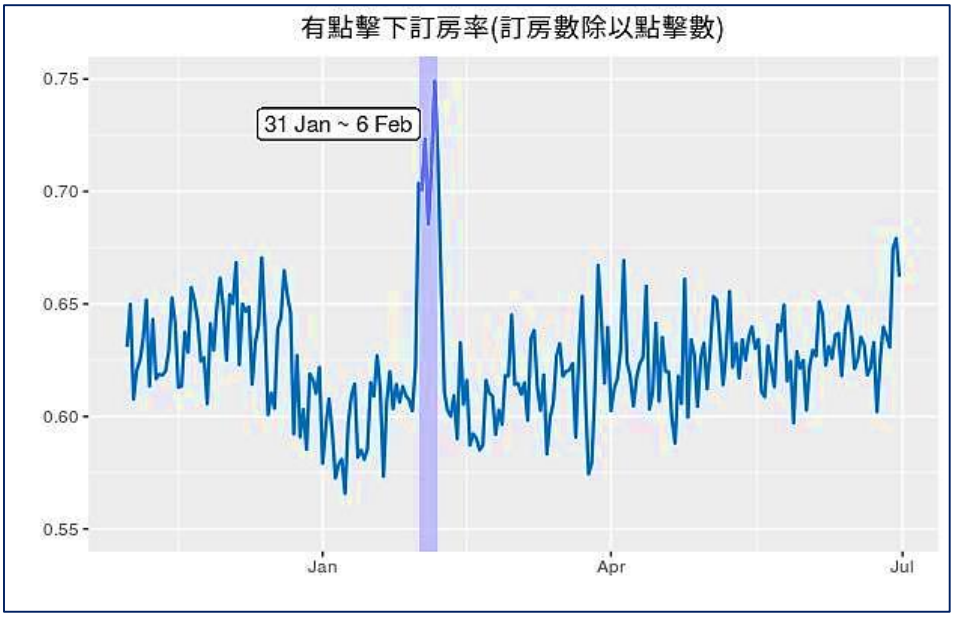


(AUC = 81.05%)



三、 干擾因子 — R

除了 Expedia 提供的資料之外，可能有其他外部因子干擾，造成訂房率變化，我們希望能找出外部因素，解釋訂房率變化。



圖中橫軸為每一日，縱軸為**有點擊下的訂房率**，可見大部份維持在 0.6 至 0.65，唯 **1 月 31 日至 2 月 6 日**期間飆升至 0.7 附近，最高至 0.75，然後又跌至原來水平。我們相信在這段時間附近，應該有突發事件導致訂房率上升。

我們使用 **R 的 CausalImpact** 套件，去計算可能事件的影響有多大。計算沒有事件下的依變項會怎樣發展，對比實際有發生事件的情況，推測事件的影響力有多大。套件需要三個參數，分別是**依變項**、**自變項**及**時間切割點**。

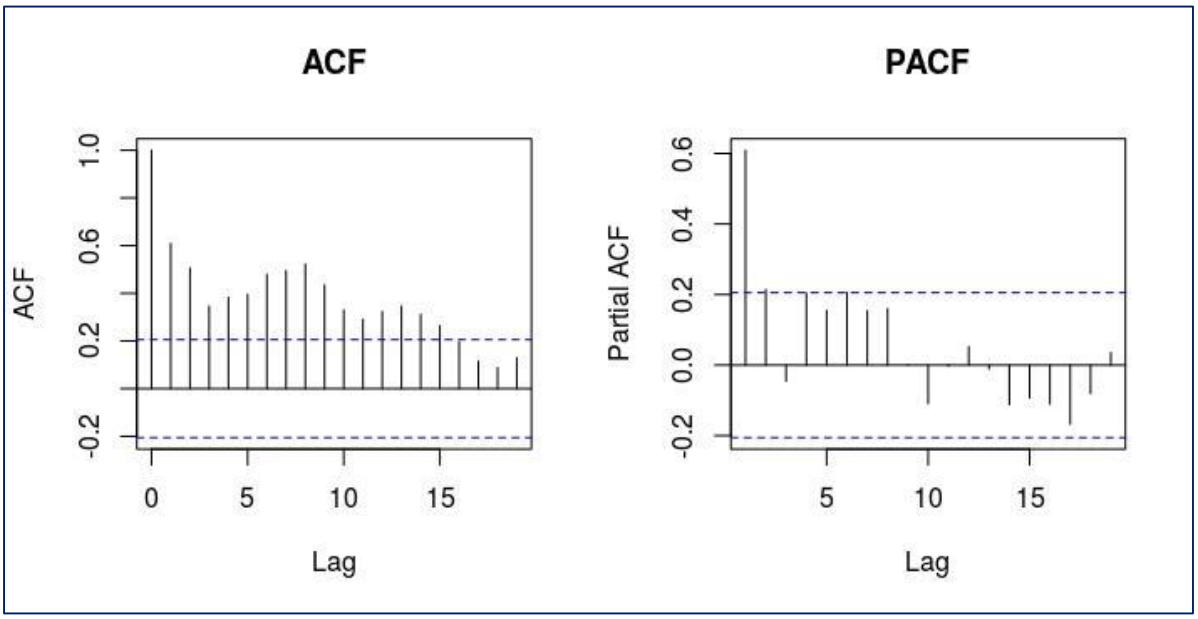
有關 CausalImpact 的自變項選擇，我們採用 **ARIMA(Autoregressive Integrated Moving Average model)**模型配適，配適結果為 **ARIMA(5,0,5)**。老師及助教曾建議 Google correlate 及尋找經濟指標等方法，我們皆已嘗試，Google Correlate 最相關的是「how to be a surgon」還有「garlic hair treatment」，唯結果難於說明與訂房率間的關係或配適情況欠佳，故採用 ARIMA 模型尋找配適值使其為 CausalImpact 套件計算下的自變項。

ARIMA 模型常用於時間序列分析，特點是可以**不需要自變數**，僅以依變數自身的前期去預測下一期。參數有三個，分別是**自我迴歸期數(p)**、**移動平均的期數(q)**及使數列符合平穩狀態的**差分次數(d)**。有關技術細節，受篇幅限制，不會詳細說明。

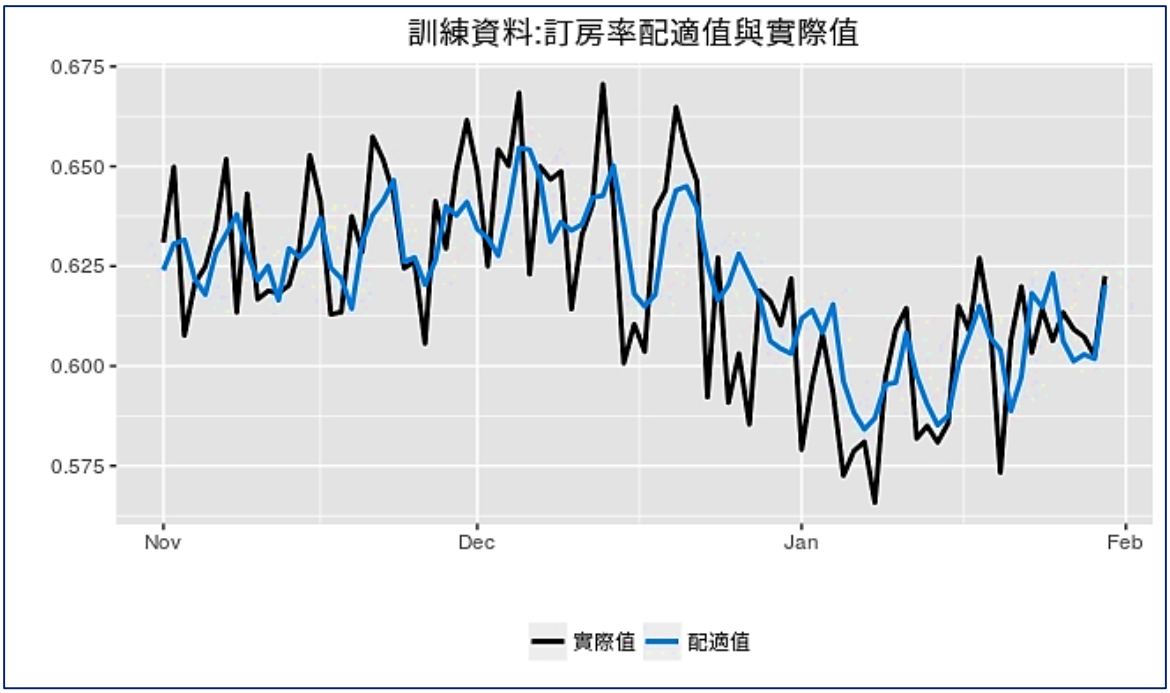
$$\left(1-\sum_{i=1}^{p}\phi_i L^i\right)Y_t = \left(1+\sum_{i=1}^{q}\theta_i L^i\right)\varepsilon_t\,.$$

(圖片來源：維基百科)

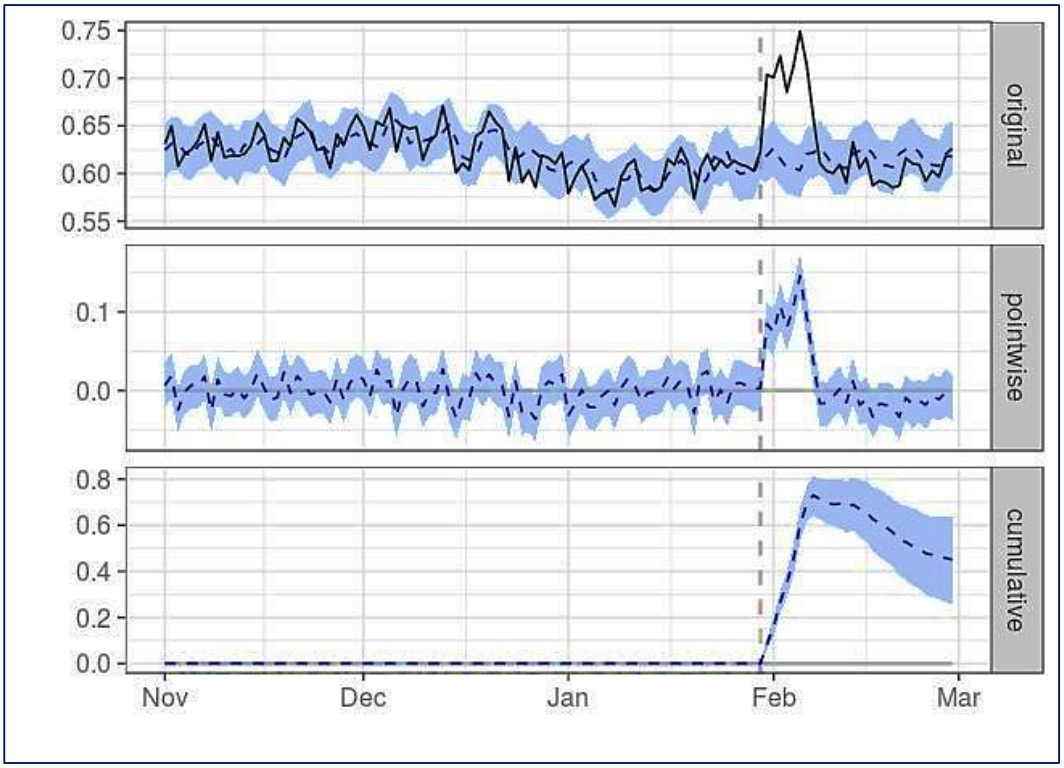
我們選取訓練資料的時間點為 **2012 年 11 月 1 日至 2013 年 1 月 30 日**，測試資料集為 **2013 年 1 月 31 日至 2013 年 2 月 28 日**。首先要決定差分次數(d)，為得知數列是否符合平穩狀態，我們使用 **Phillips-Perron 單根檢定**，顯示 p-value 為 0.01，拒絕數列為單根的虛無假設，所以設定 d 為 0。其次我們要決定 p 和 q，但較難以 ACF 及 PACF 圖判斷 p 與 q 的值，所以用 **AIC(Akaike information criterion)**為標準，嘗試不同 p 與 q 的組合，選擇 AIC 最低者，最後得出 p 與 q 都為 5。所以模型為 **ARIMA(p,d,q) = ARIMA(5,0,5)**。



以 ARIMA(5,0,5)模型得出配適值後：



以此為 CausalImpact 的自變數，計算結果如下圖：

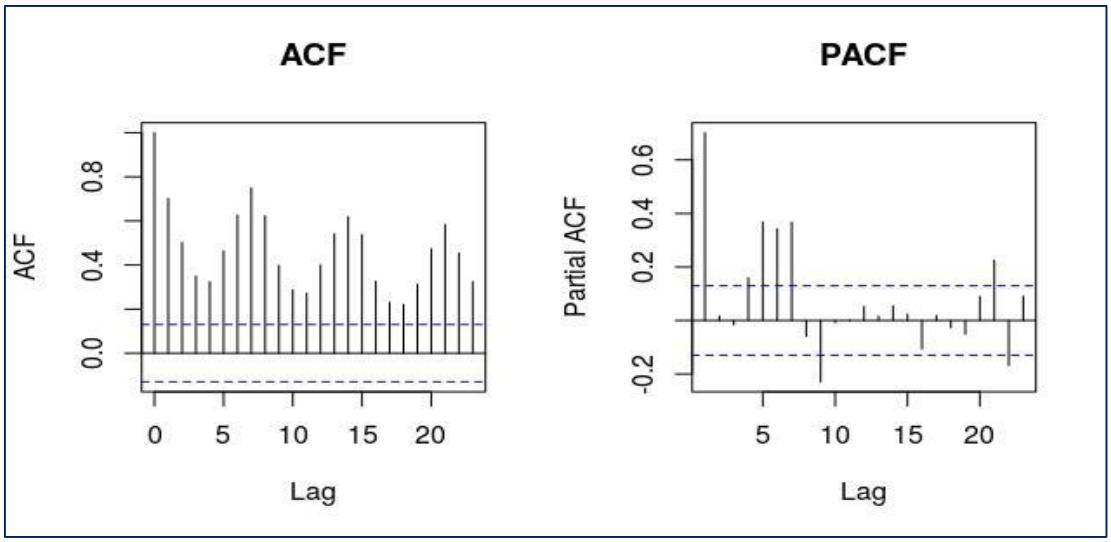


可見在 **2013 年 1 月 31 日開始突然偏離預測，2 月 6 日之後回到預測範圍**。我們推測未必是 1 月 31 日的事件，由於資料記錄的是訂房日期，可能真正的事件在入住日期，已知資料是美國資料為大宗（Expedia 發跡自美國），二月初前後，能提升訂房率的日子，推測最大可能是**情人節**。有點擊下的訂房率平均為 0.621，高峰期七天平均 0.713，相差 0.092。所以如果有人在 1 月 31 日至 2 月 6 日期間點擊，平均會比日常情況的訂房率高 10%。

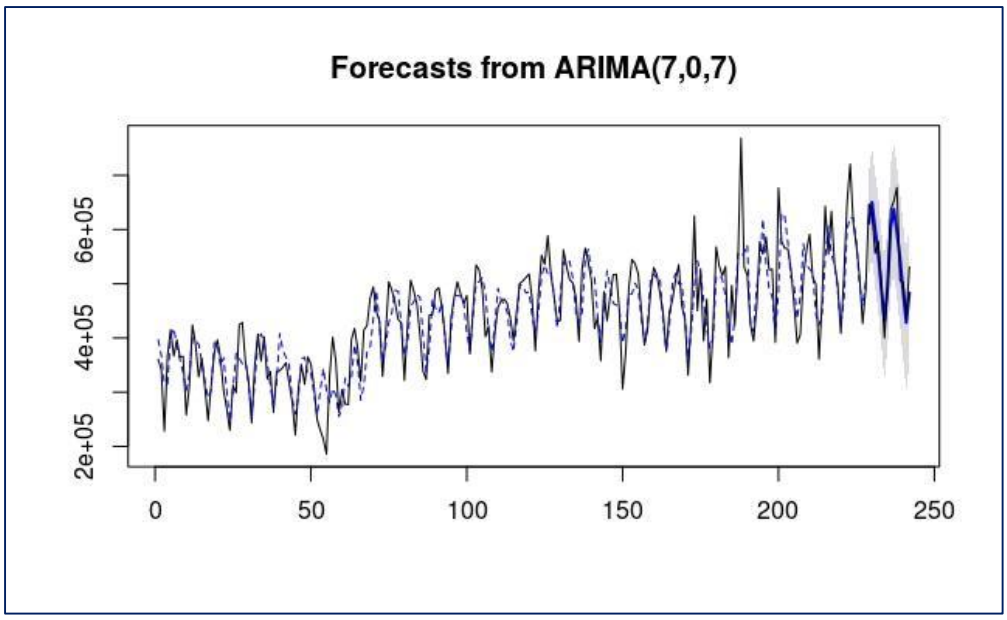
四、 營收預測

同樣，我們也可以用 **ARIMA 模型**預測每日營收。由於資料僅記錄訂房的美元價格(gross_bookings_usd)，我們無從得知 Expedia 從中能獲多少營收，所以**視訂房價格為營收計算**、預測 Expedia 營收。雖然原資料有一千萬筆，記錄多種不同欄位，但預測營收可以不需要知道每筆資料是否有訂房，只需要過往的每日營收，就可以預測往後的營收。

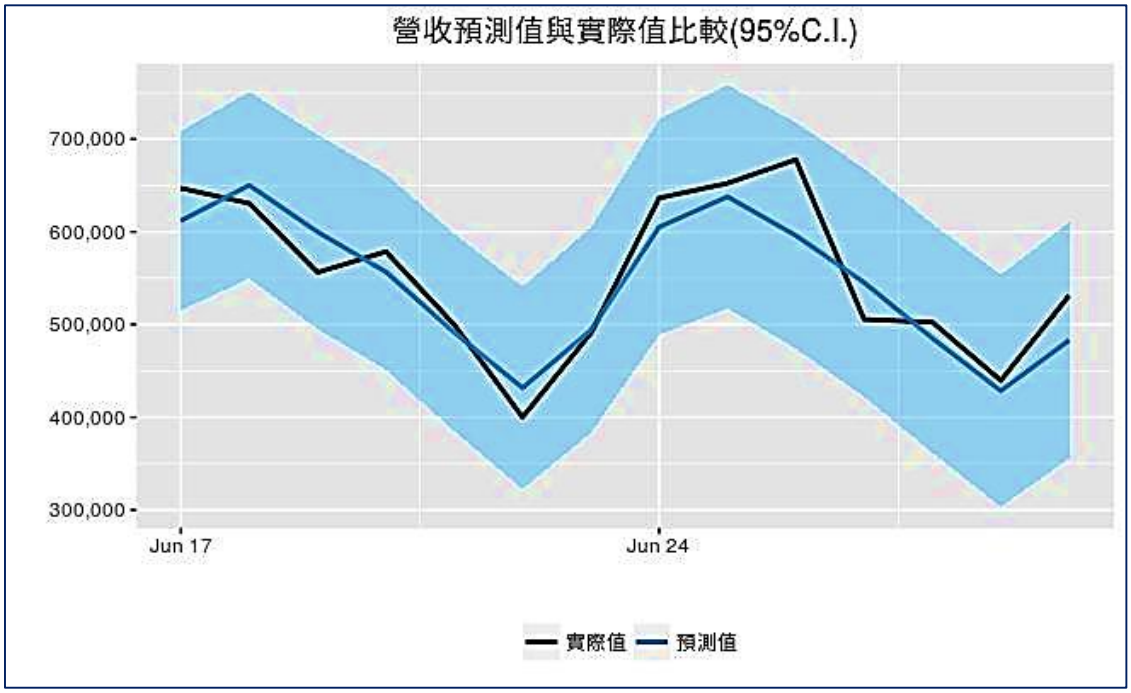
以**最後兩週為預測資料**，**其餘為訓練資料**。Phillips-Perron 單根檢定的 p-value 為 0.01，拒絕單根的虛無假設，數列符合穩定條件。ACF 及 PACF 圖看數出列週期為 7 天，所以判斷模型選用 **ARIMA(7,0,7)**。



可以看出，配適值（藍色虛線）與實際資料值（黑色實線）接近。資料後期的藍色實線是預測值，淺藍色是預測值的 **90%信賴區間**。由於我們目標是預測是否準確，所以只需對比預測值與測試資料是否接近。



結果顯示，兩者相當接近。**MAPE(Mean Absolute Percentage Error)**為 5.176%，**RMSE(Root Mean Square Error)**為 35115.25。說明可以用 ARIMA(7,0,7)預測 Expedia 每日營收。



五、 總結

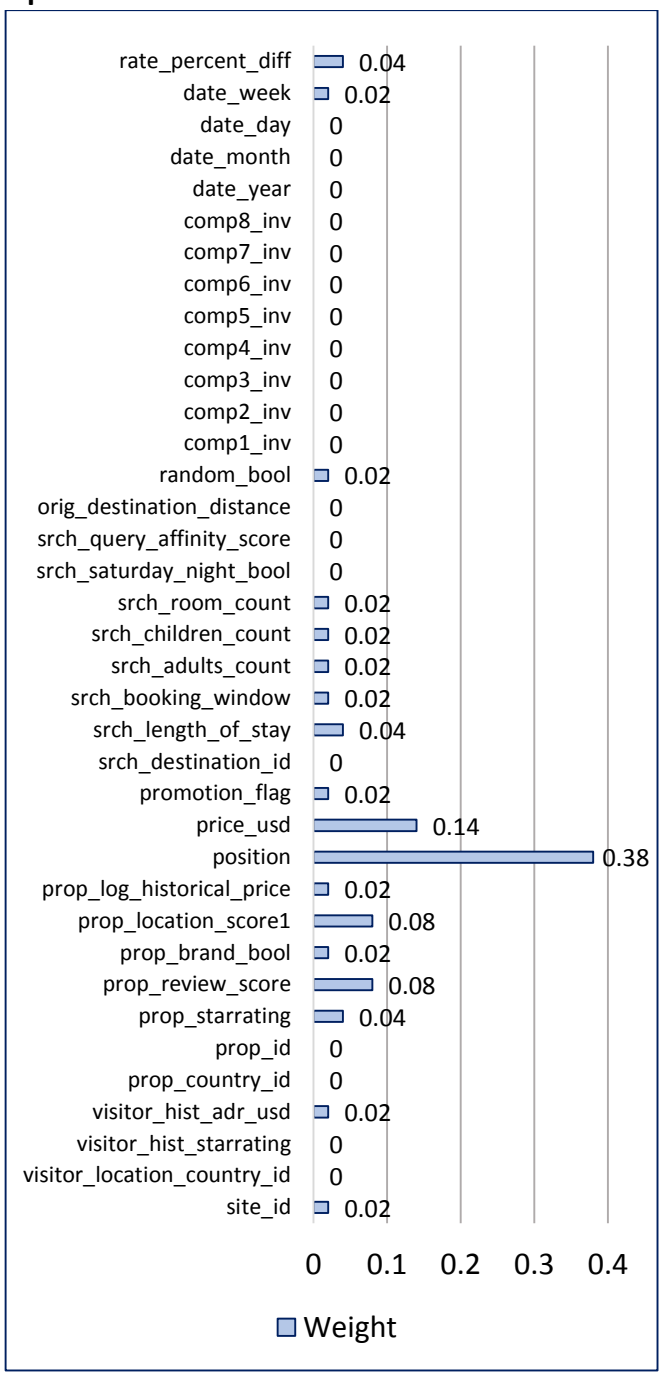
我們透過 **AWS-S3 儲存資料**、**GCP 雲端運算平台**及 **PySpark**，預測 Expedia 的**點擊率**。在 Training data 用重複抽樣的方法使 Click、not Click 等量，改善資料的不對稱問題，主觀處理及挑選特徵值(包括 **semi-supervised learning**)；模型使用 **Decision tree 的 AdaBoost** 搭配 **5-fold CV**。測試資料集顯示比較重要的特徵值包括**飯店地點**、**瀏覽時的報價**，而我們彙整的 rate_percent_diff(競爭者之間的價格競爭)也有一定的解釋力，**AUC = 81.05%**。

透過 **R 的 CausalImpact** 套件，試圖找出具影響力的干擾因子。我們發現**有點擊下的訂房率**，在 **1 月 31 日至 2 月 6 日**期間升超過 **10%**。之後嘗試以 **Google correlate** 結果及**經濟指標**追蹤訂房率變化，唯狀況欠佳，故採用 **ARIMA 模型**配適。我們認為干擾因子是**情人節**，雖然訂房率變化是在 1 月 31 日至 2 月 6 日，但訂房日期不等同入住日期。策劃情人節活動必須萬無一失，也必然提早訂房，才使得 1 月 31 日至 2 月 6 日有點擊下的訂房率，平均比其他時間點高 10%。

預測營收的部分，由於**每日營收有規律的路徑**，因而使用 **ARIMA 模型** (用過往營收預測未來營收)。測試資料顯示 **MAPE = 5.176%**，**RMSE = 35115.25**，預測效果不俗。

六、 附錄

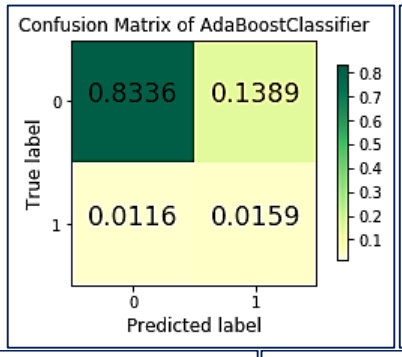
附圖(2.1) **Feature Importance**

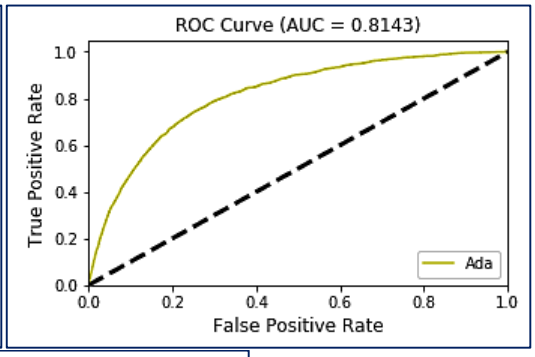


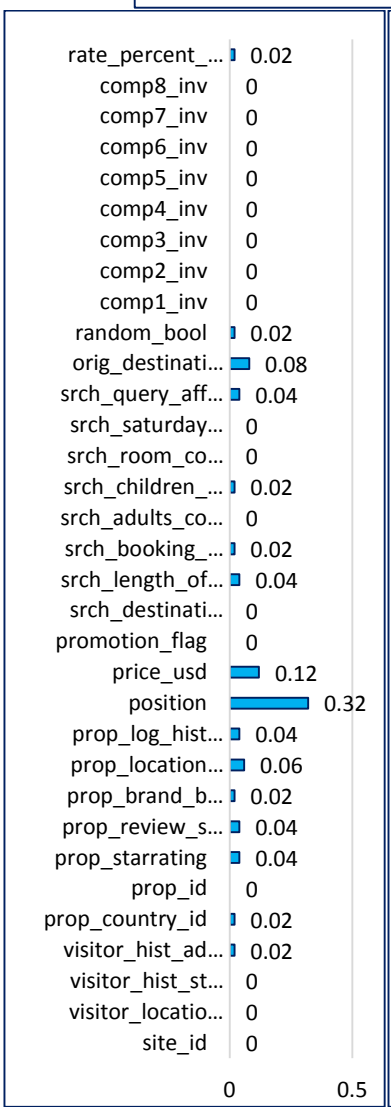
附圖(2.2) **將 date_month、date_day、date_week 轉為 Dummy variable 的模型表現 (整體資料量：20 萬筆，本地端運算)**

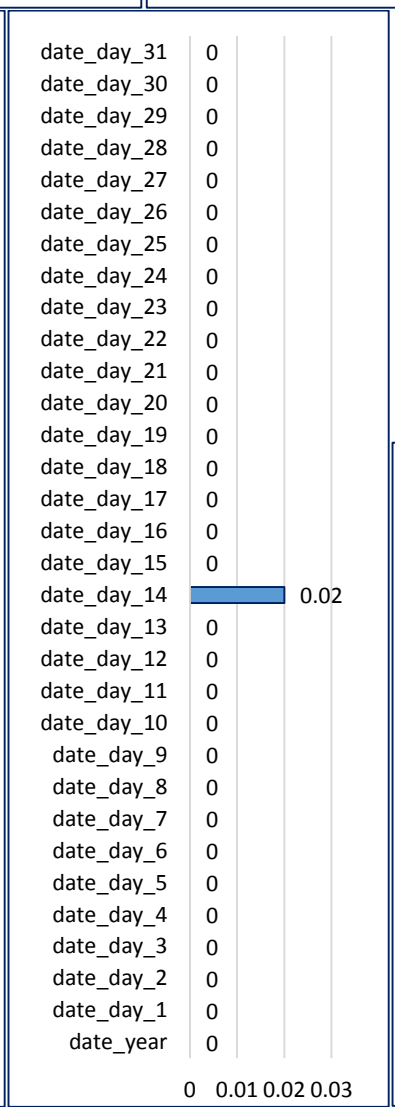
AdaBoostClassifier report

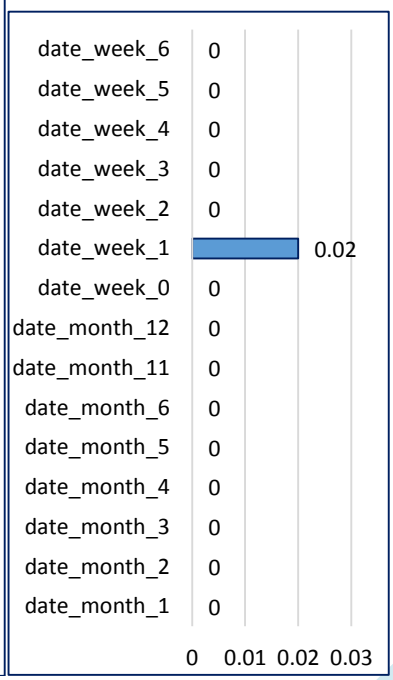
	precision	recall	f1-score	support
not Click	0.9865	0.8476	0.9118	77657
Click	0.1008	0.5948	0.1724	2231
avg / total	0.9617	0.8406	0.8911	79888



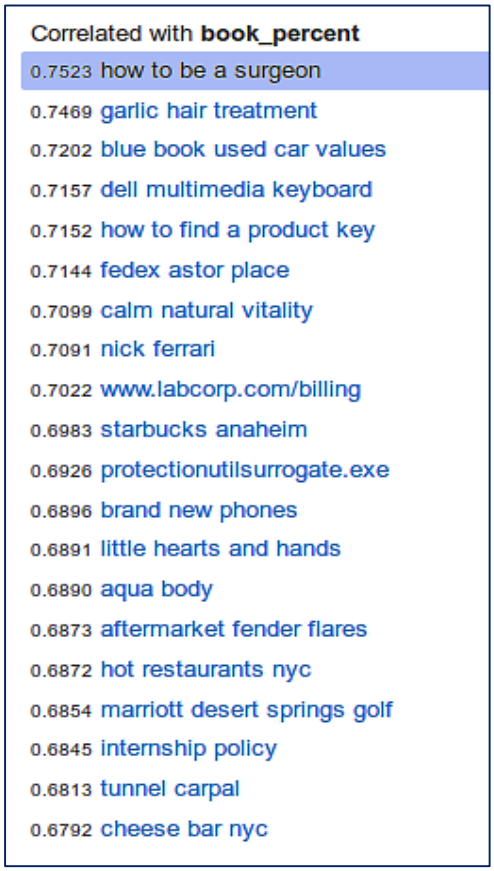








附圖(3.1) **Google Correlate 結果**



附表(3.1) **ARIMA 預測營收結果**

日期	資料值	配適值	日期	資料值	配適值
2013-06-17	646981.66	611894.5789	2013-06-24	636471.97	605117.9263
2013-06-18	630629.30	650209.9154	2013-06-25	652277.19	637827.0342
2013-06-19	556163.65	599826.2003	2013-06-26	677723.33	595971.3798
2013-06-20	578599.23	556743.7171	2013-06-27	505529.48	545238.3025
2013-06-21	499828.76	491455.4771	2013-06-28	502407.42	485122.7637
2013-06-22	400153.46	431700.1440	2013-06-29	439591.95	428513.5838
2013-06-23	490519.56	493808.2320	2013-06-30	531130.52	483017.2889