數據科學與
大數據分析--
期末報告

資科三　　蔡漢龍
金融碩一 沈柏宇
統計碩二 梁家安

分工

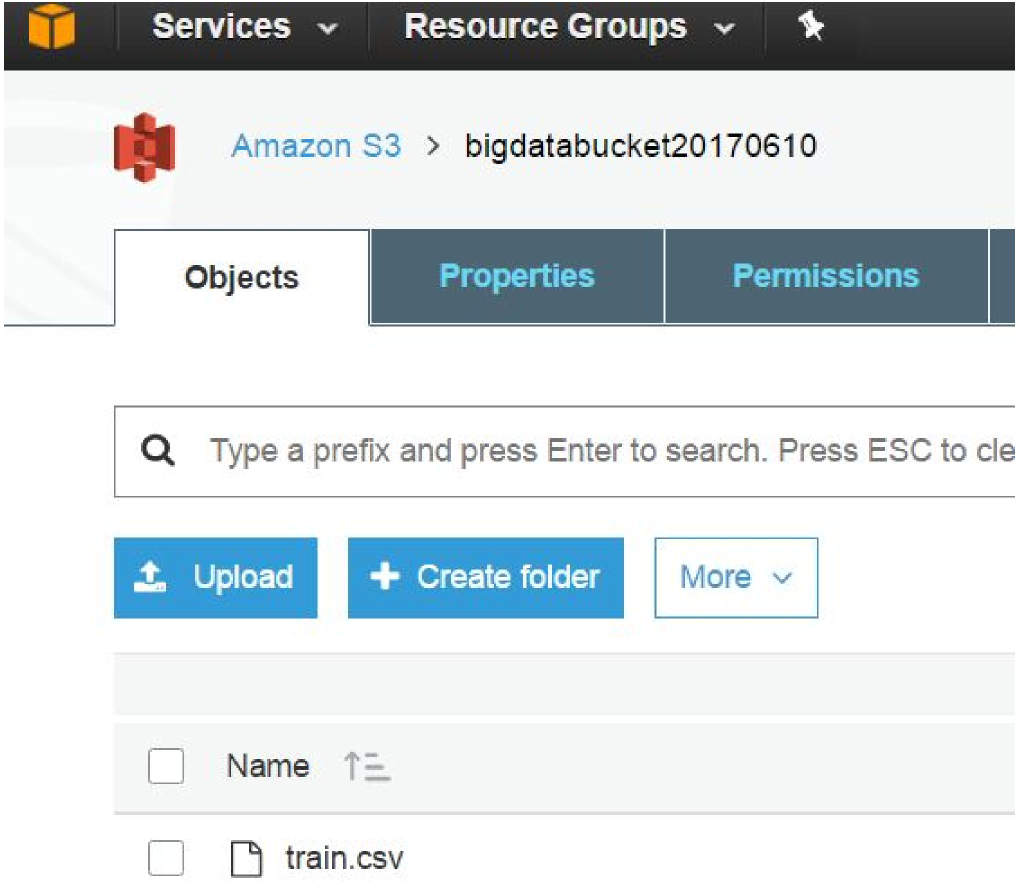
- 蔡漢龍
  - AWS
  - GCP
- 沈柏宇
  - Python
  - 分類建模
- 梁家安
  - R
  - 干擾因子

大綱

- 環境建置
  - AWS與GCP
- 點擊率預測 – AdaBoostClassifier
  - 資料處理
  - 方法與結果 (Feature importance、Confuscion matrix、ROC、Precision report)
- 干擾因子
  - 方法 (ARIMA、BSTS)
  - 結果
- 營收預測
  - 方法 (ARIMA)
  - 結果

環境建置

AWS-S3


Services
Resource Groups
Amazon S3 > bigdatabucket20170610
Objects
Properties
Permissions
Type a prefix and press Enter to search. Press ESC to cle
Upload
Create folder
More
Name

□ train.csv

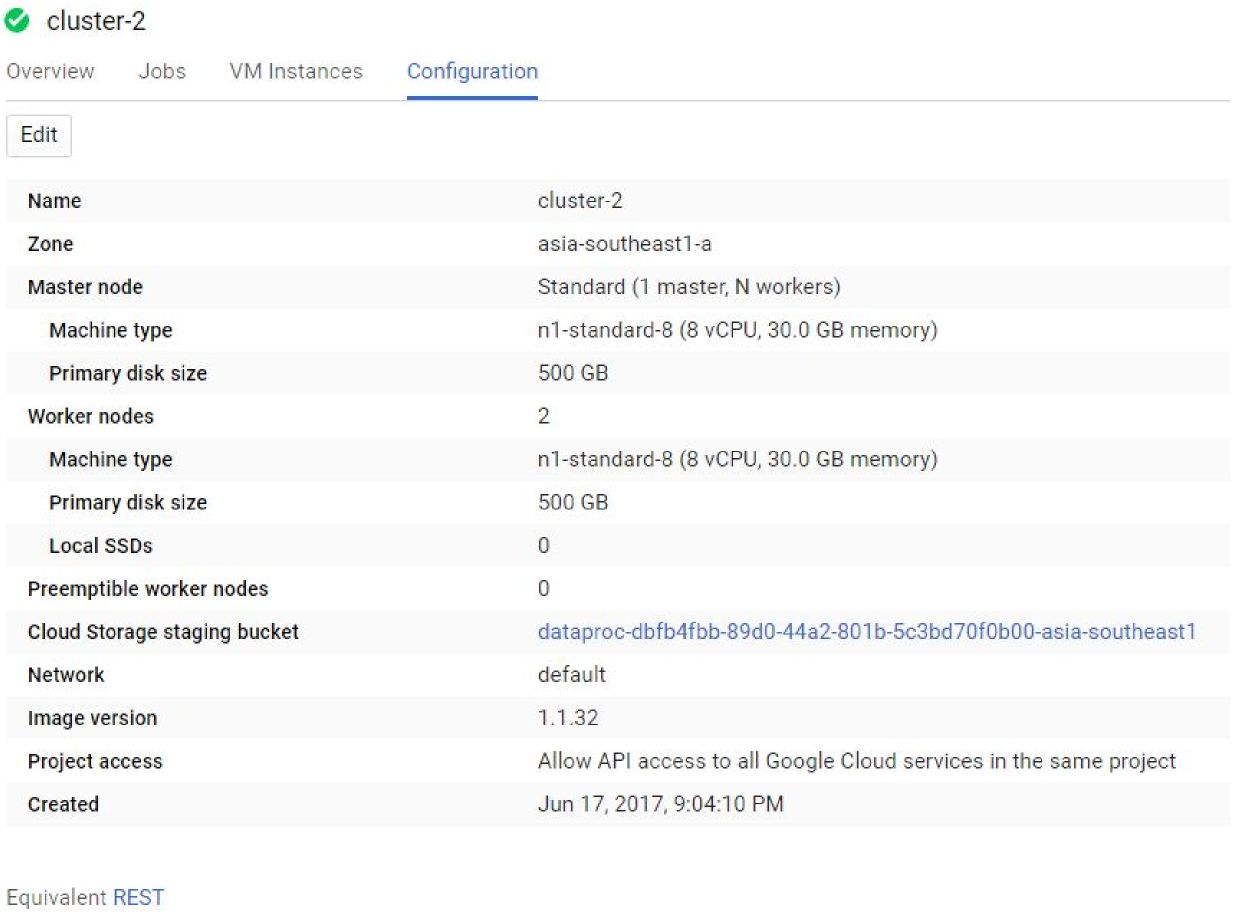
環境建置

AWS-S3

```
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'])
```

環境建置

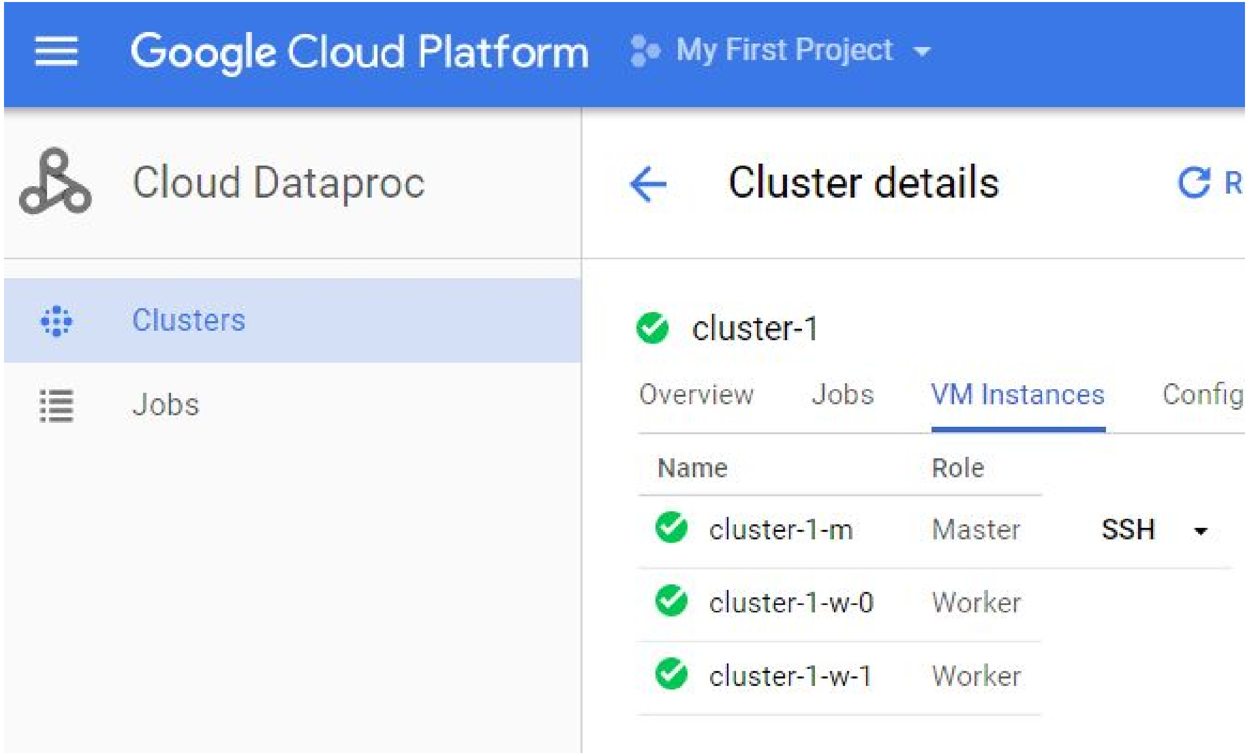
GCP - Dataproc



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

環境建置

GCP - Dataproc



Name	Role
cluster-1-m	Master
cluster-1-w-0	Worker
cluster-1-w-1	Worker

環境建置

GCP - Dataproc

```
 export AWS_ACCESS_KEY_ID= XXXXXXXXXXX
 export AWS_SECRET_ACCESS_KEY= XXXXXXXXXXXX
spark-submit Demo.py
```

點擊率預測

資料處理

date_time → 分拆成 date_year, date_month, date_day

visitor_hist_starrating → nan設為0 (仿效prop_starrating)

visitor_hist_adr_usd → nan設為0

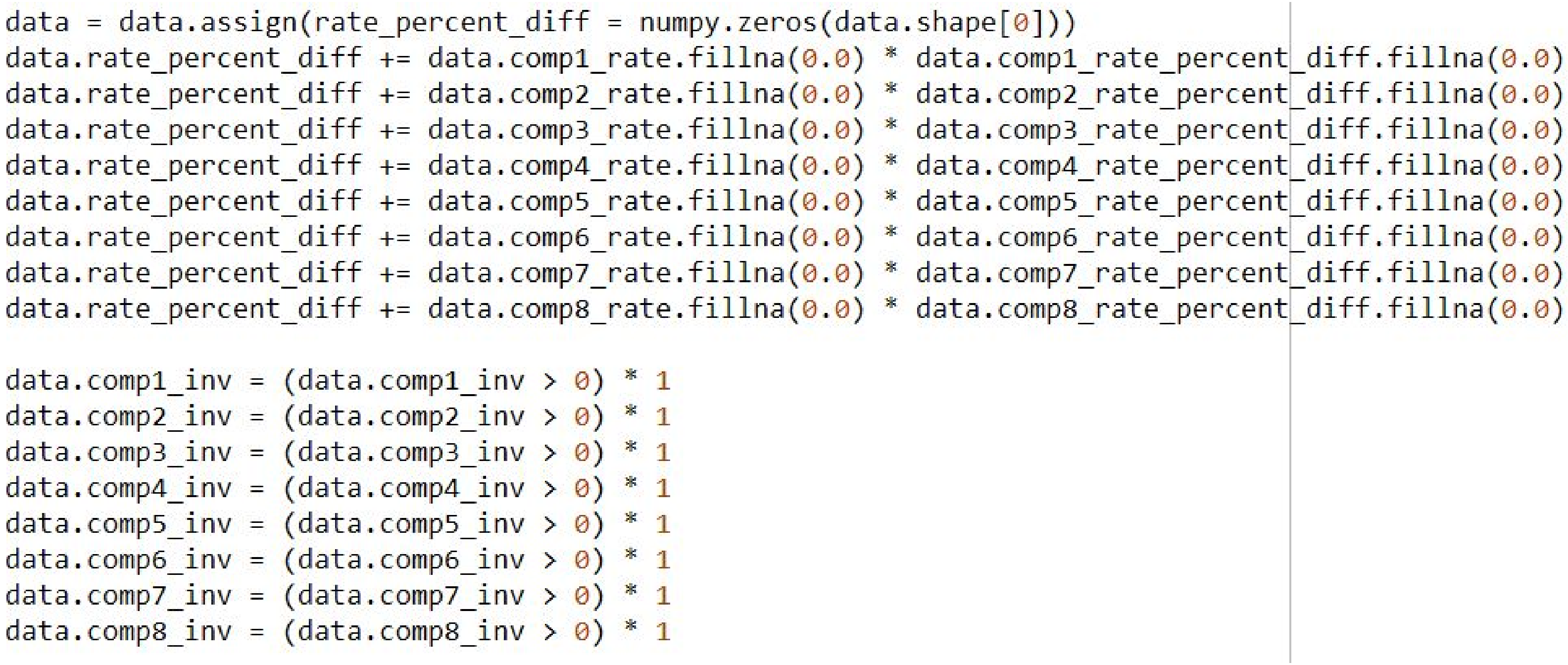
prop_location_score2 → nan設為0，與prop_location_score1加總

srch_query_affinity_score → 轉成機率，null設為0

orig_destination_distance → 用site_id, visitor_location_country_id, prop_country_id, srch_destination_id做分群(K-means)，計算群集的平均 值預測nan

點擊率預測

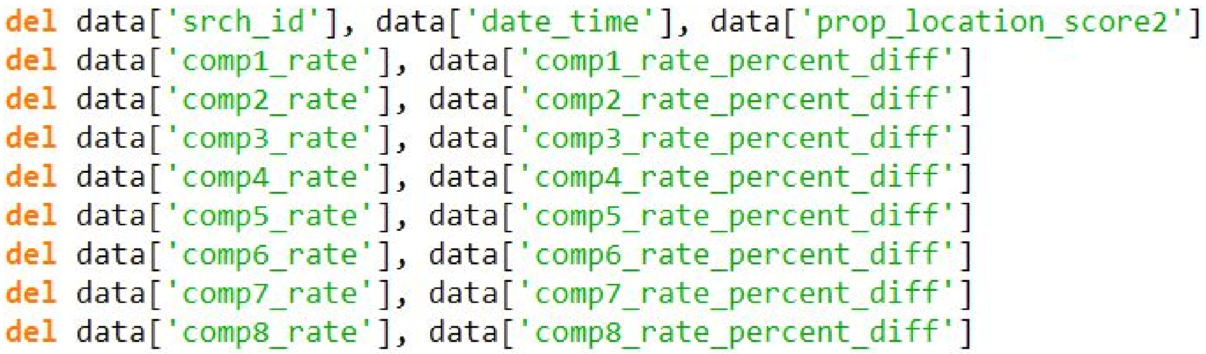
資料處理



```
data = data.assign(rate_percent_diff = numpy.zeros(data.shape[0]))
data.rate_percent_diff += data.comp1_rate.fillna(0.0) * data.comp1_rate_percent_diff.fillna(0.0)
data.rate_percent_diff += data.comp2_rate.fillna(0.0) * data.comp2_rate_percent_diff.fillna(0.0)
data.rate_percent_diff += data.comp3_rate.fillna(0.0) * data.comp3_rate_percent_diff.fillna(0.0)
data.rate_percent_diff += data.comp4_rate.fillna(0.0) * data.comp4_rate_percent_diff.fillna(0.0)
data.rate_percent_diff += data.comp5_rate.fillna(0.0) * data.comp5_rate_percent_diff.fillna(0.0)
data.rate_percent_diff += data.comp6_rate.fillna(0.0) * data.comp6_rate_percent_diff.fillna(0.0)
data.rate_percent_diff += data.comp7_rate.fillna(0.0) * data.comp7_rate_percent_diff.fillna(0.0)
data.rate_percent_diff += data.comp8_rate.fillna(0.0) * data.comp8_rate_percent_diff.fillna(0.0)

data.comp1_inv = (data.comp1_inv > 0) * 1
data.comp2_inv = (data.comp2_inv > 0) * 1
data.comp3_inv = (data.comp3_inv > 0) * 1
data.comp4_inv = (data.comp4_inv > 0) * 1
data.comp5_inv = (data.comp5_inv > 0) * 1
data.comp6_inv = (data.comp6_inv > 0) * 1
data.comp7_inv = (data.comp7_inv > 0) * 1
data.comp8_inv = (data.comp8_inv > 0) * 1
```

點擊率預測

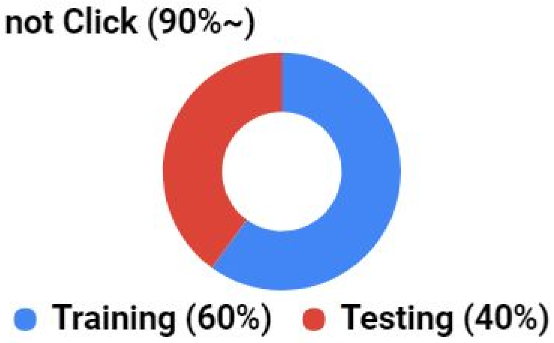
資料處理

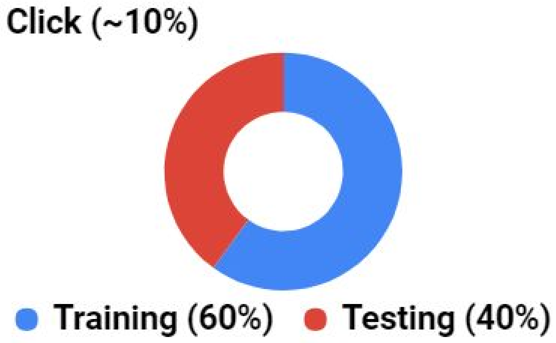


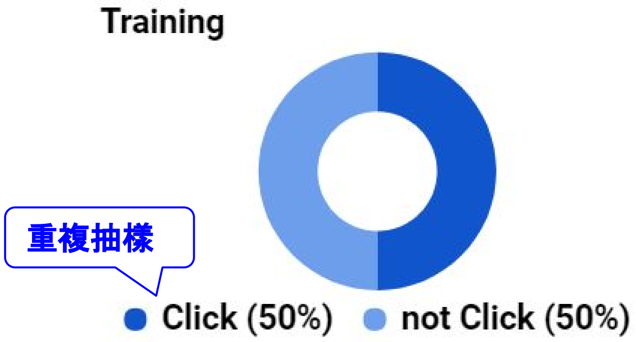
```
del data['srch_id'], data['date_time'], data['prop_location_score2']
del data['comp1_rate'], data['comp1_rate_percent_diff']
del data['comp2_rate'], data['comp2_rate_percent_diff']
del data['comp3_rate'], data['comp3_rate_percent_diff']
del data['comp4_rate'], data['comp4_rate_percent_diff']
del data['comp5_rate'], data['comp5_rate_percent_diff']
del data['comp6_rate'], data['comp6_rate_percent_diff']
del data['comp7_rate'], data['comp7_rate_percent_diff']
del data['comp8_rate'], data['comp8_rate_percent_diff']
```

點擊率預測

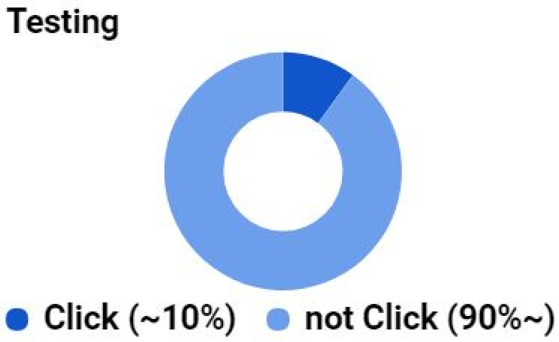
資料處理







重複抽樣

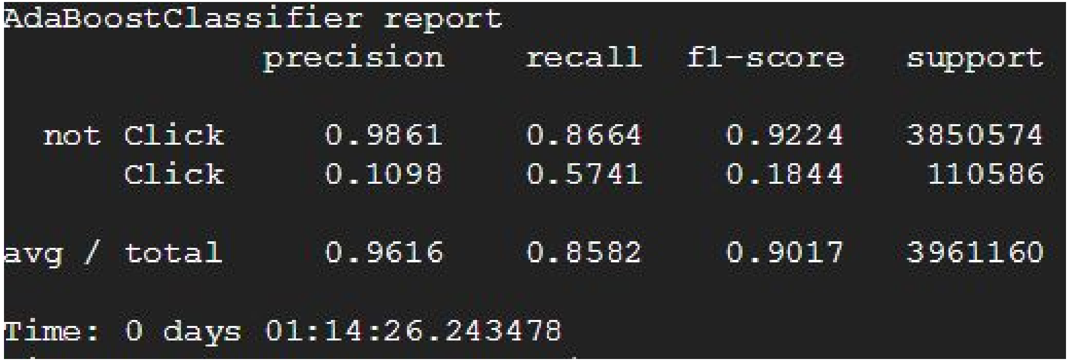


點擊率預測

方法與結果

```
Ada = sklearn.model_selection.GridSearchCV( sklearn.ensemble.AdaBoostClassifier(),
                                            { 'n_estimators': [10, 20, 50], 'learning_rate': [0.01, 0.5, 1] },
                                            cv=5 ).fit(x_train, y_train).best_estimator_
```

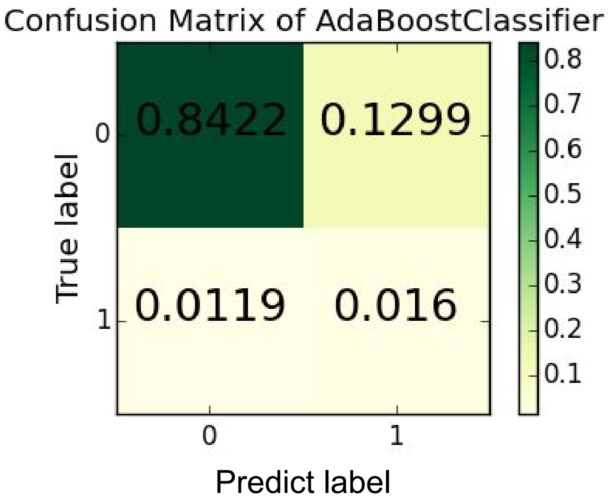
AdaBoostClassifier parameters with cross-validation:
{ n_estimators: 50, base_estimator: None, random_state: None, learning_rate: 1.0, algorithm: SAMME.R}

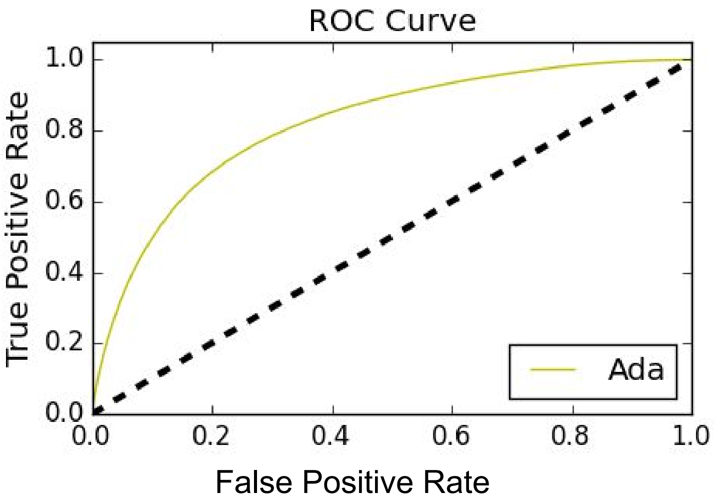


```
AdaBoostClassifier report
             precision    recall  f1-score   support

  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
```

點擊率預測

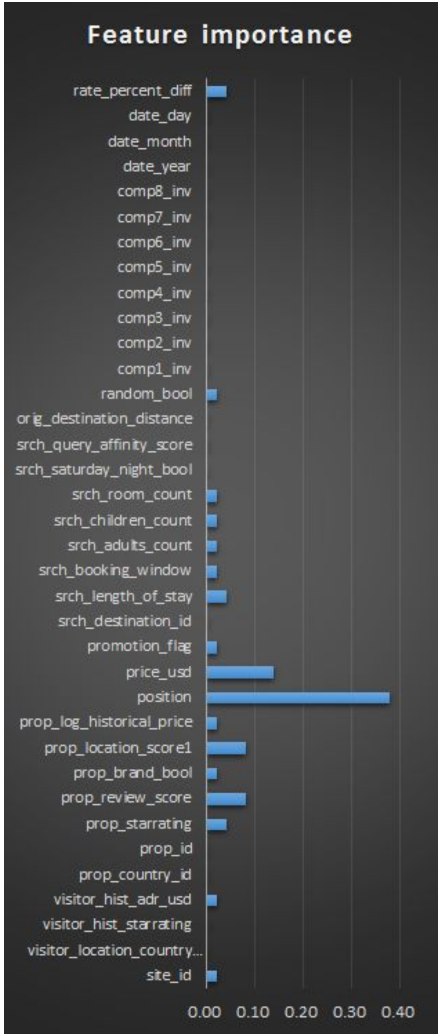
方法與結果


Confusion Matrix of AdaBoostClassifier
0.8422
0.1299
0.0119
0.016
0
1
True label
0
1
Predict label
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1

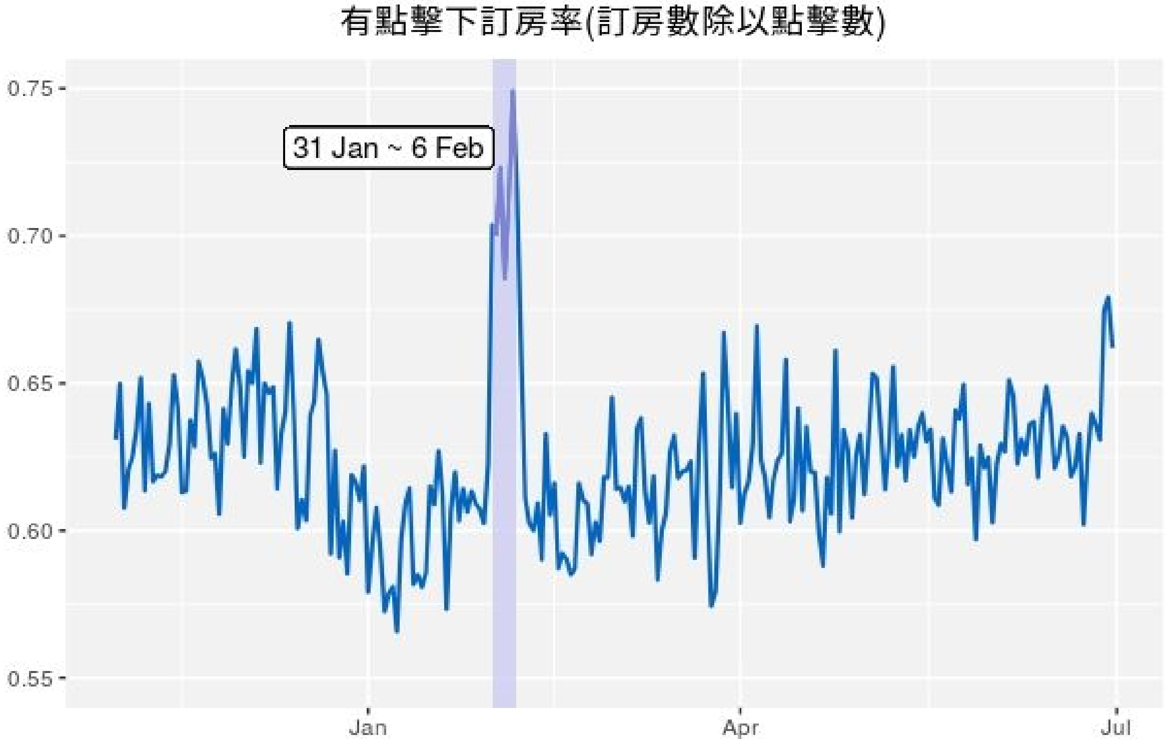

ROC Curve
True Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Ada

點擊率預測

方法與結果


Feature importance
rate_percent_diff
date_day
date_month
date_year
comp8_inv
comp7_inv
comp6_inv
comp5_inv
comp4_inv
comp3_inv
comp2_inv
comp1_inv
random_bool
orig_destination_distance
srch_query_affinity_score
srch_saturday_night_bool
srch_room_count
srch_children_count
srch_adults_count
srch_booking_window
srch_length_of_stay
srch_destination_id
promotion_flag
price_usd
position
prop_log_historical_price
prop_location_score1
prop_brand_bool
prop_review_score
prop_starrating
prop_id
prop_country_id
visitor_hist_adr_usd
visitor_hist_starrating
visitor_location_country...
site_id
0.00
0.10
0.20
0.30
0.40

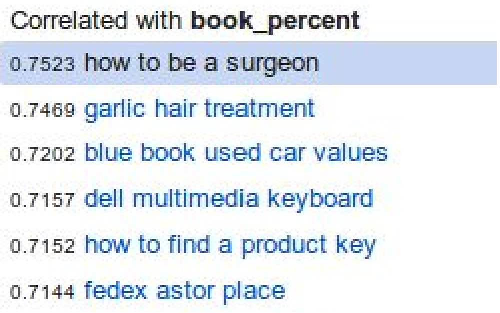
干擾因子



- R>CausalImpact
- 三個參數
  - 依變項（Y）
  - 自變項（X）
  - 時間切割點

干擾因子

- 自變項(X)選擇
  - ~~google correlate~~
    - 難以說明



Correlated with **book_percent**

0.7523	how to be a surgeon
0.7469	garlic hair treatment
0.7202	blue book used car values
0.7157	dell multimedia keyboard
0.7152	how to find a product key
0.7144	fedex astor place

  - ~~經濟指標~~
    - 配適欠佳
  - ARIMA

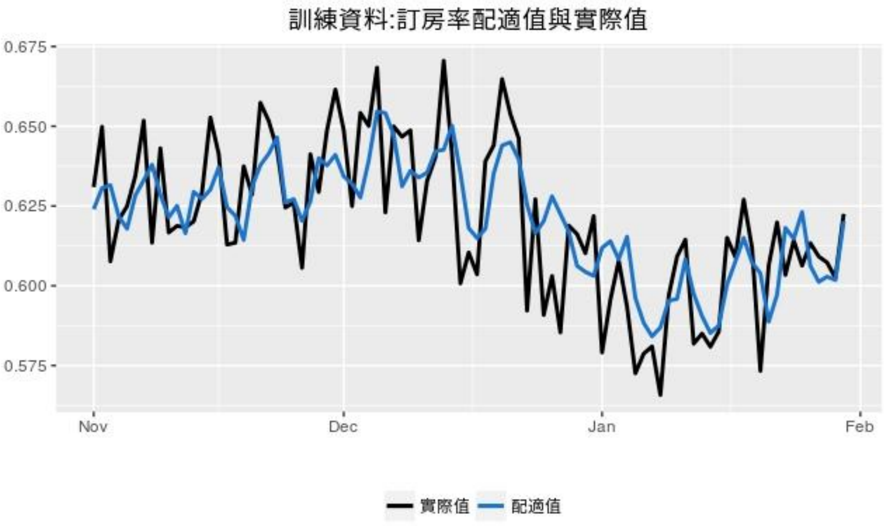
- ARIMA(Autoregressive Integrated Moving Average model)
  - 時間序列模型
  - 不需要自變項(X)
  - 自己過去預測自己未來
  - 參數
    - 自我迴歸期數(p)
    - 移動平均期數(q)
    - 使數列平穩的差分次數(d)

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

*公式引用維基百科

干擾因子

- 訓練資料
  - 2012/11/01～2013/01/30
- 測試資料
  - 2013/01/31～2013/02/28

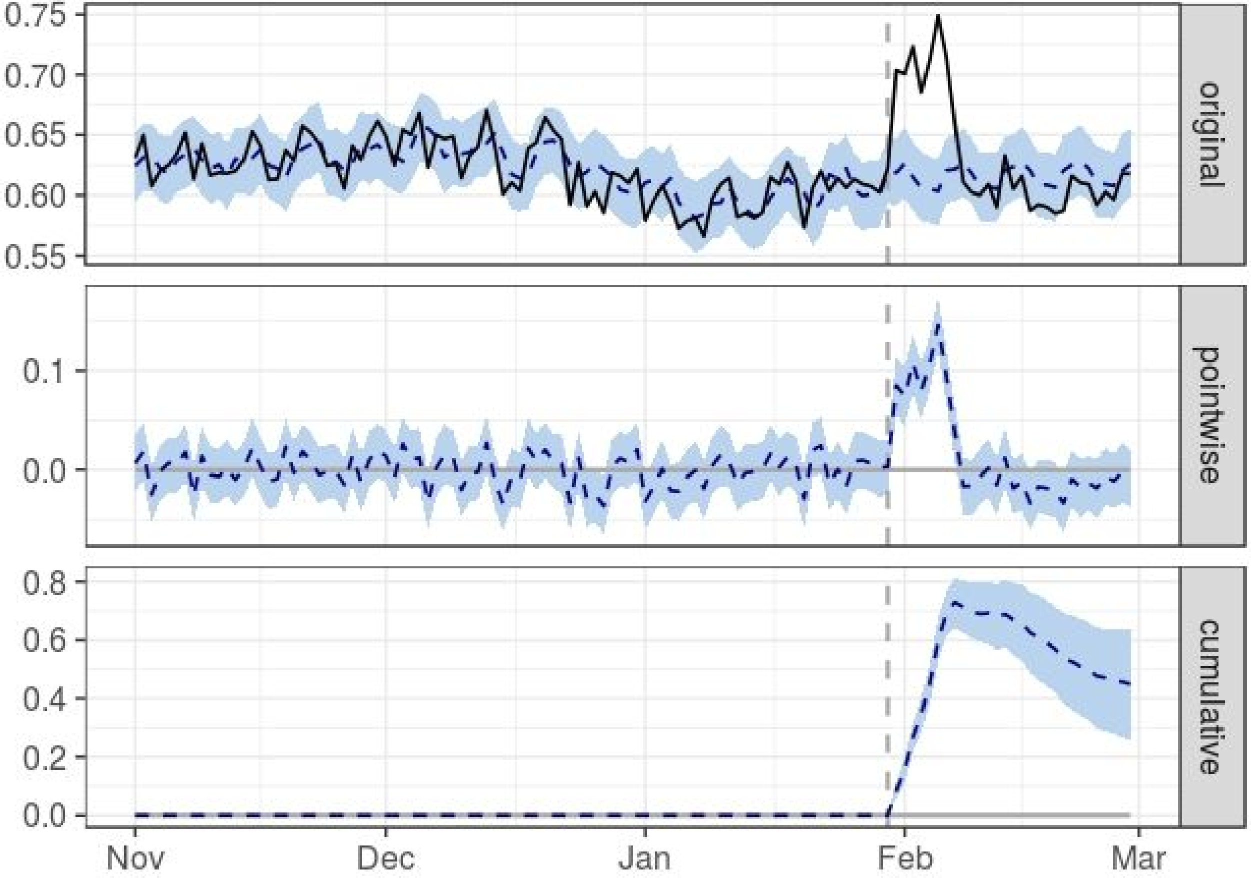


- R > library(forecast)
- 決定差分次數(d)
  - Phillips-Perron Unit Root Test
    - pvalue=0.01
      - 數列平穩
        - d=0
- 決定p與q的值
  - 暴力法測試
    - 選AIC (Akaike information criterion) 最低者
      - p=q=5
- **ARIMA(5,0,5)**

干擾因子

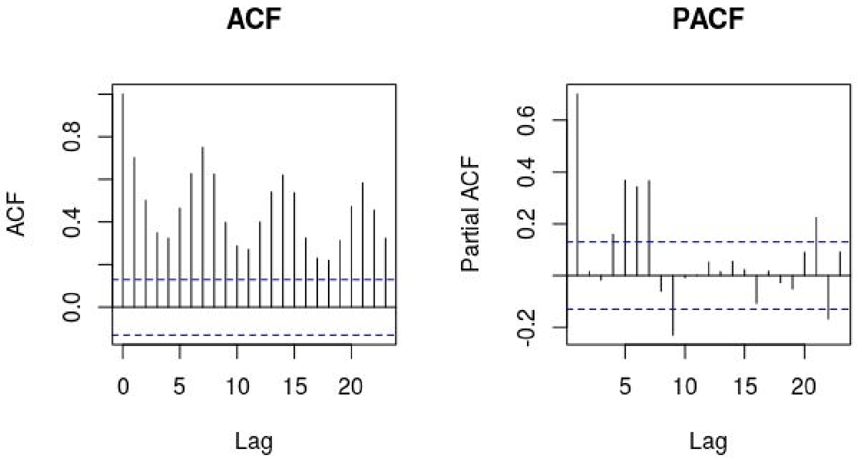
- 01/31開始偏離預測
- 02/06回到預測範圍
- 訂房日期 ≠入住日期
- 情人節！

- 之前平均0.621
- 高峰期七天平均0.713
- 相差0.092



營收預測

- 以gross_bookings_usd代替真正營收
- 可以不需預測會否訂房
    - 用過往營收預測未來營收
        - ARIMA模型
- 測試資料:最後兩週
- ARIMA(7,0,7)



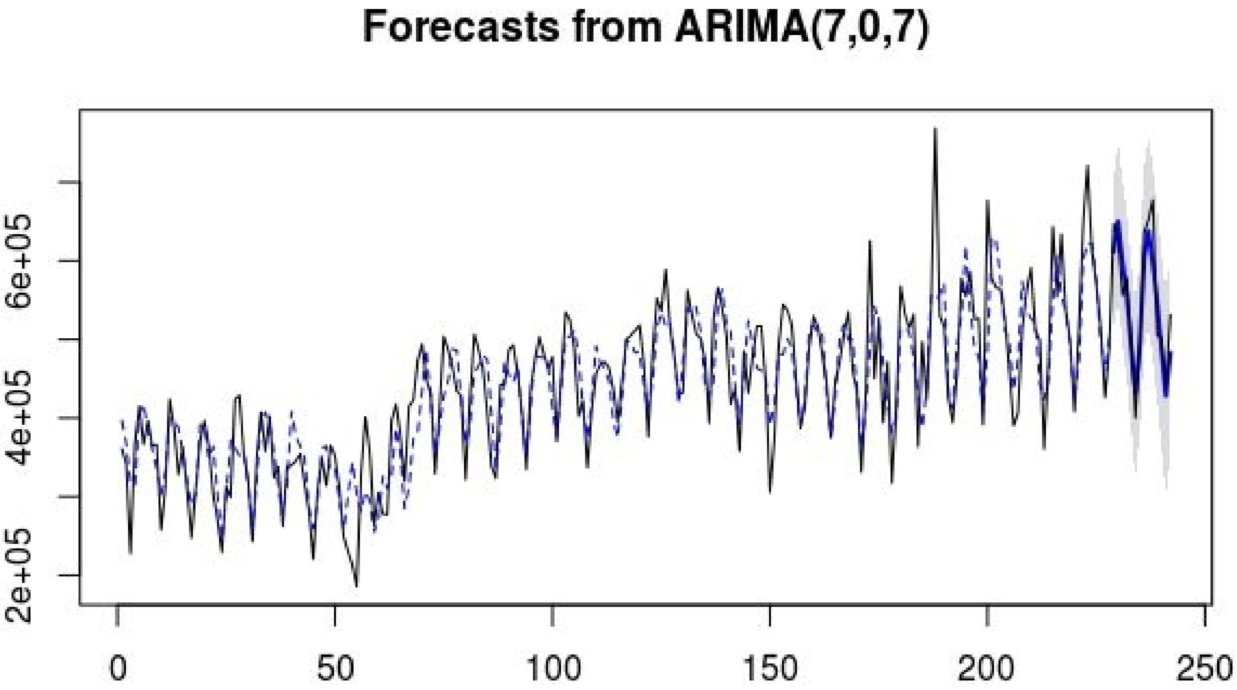
Phillips-Perron unit root test, p-value=0.01
-->符合平穩條件

營收預測

黑實線:真實資料

藍虛線:配適值

藍實線:預測值

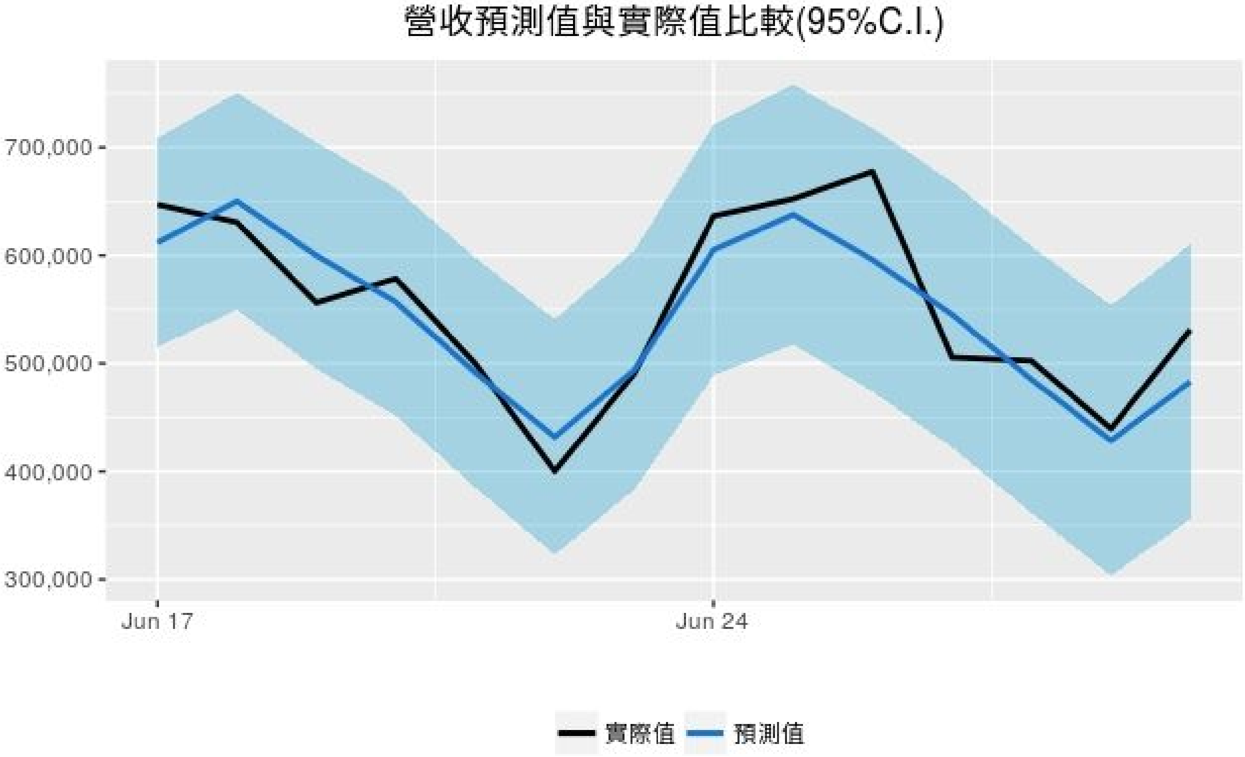


營收預測

MAPE(Mean Absolute Percentage Error)

= 5.176%

RMSE(Root Mean Square Error)
= 35115.25



github

https://github.com/BigDataAnalytics2017/ProjectOTA