

## 機器學習期末報告

競賽題目: DengAI - Predicting Disease Spread

組名: NTU\_b03901096\_ML2017finalGo

組員: B03901096 周晁德、B03901103 陳學平、B03901164 鄧惇益

分工: Preprocessing/Feature Engineering: 周晁德

Random Forest: 陳學平

LSTM: 鄧惇益

### 1. Preprocessing/Feature Engineering

#### (a) NAN data

因為一般天氣、氣候的資料具有連續性，因此我們用遺失 data 的前一筆 data 來取代這筆遺失 data，如圖(一)所示，而我們使用 pandas 中的 `pandas.fillna(method='ffill', inplace=True)`，即可輕鬆達到此效果。

weekofyear	week_start_ndvi_ne	
18	30/04/1990	0.1226
19	07/05/1990	0.1699
20	14/05/1990	0.03225
21	21/05/1990	0.1286333
22	28/05/1990	0.1962
23	04/06/1990	0.1962
24	11/06/1990	0.1129
25	18/06/1990	0.0725
26	25/06/1990	0.10245

← 原本的 missing data

#### (b) 移除一些 feature

在經過作業一後，我們可以知道當在做 training 和 testing 時，並不是把所有 feature 都用上就可以得到最好的結果，而是要選取適當的 feature，我們選擇移除 `week_start_date`、`precipitation_amount_mm`、`reanalysis_avg_tmp_k`、`reanalysis_specific_humidity_g_per_kg`，原因在 experiments and discussion 中詳述。

#### (c) 平移 data

由於當週天氣，並不會立刻影響到當週之感染人數，而是本週的感染人數較容易受到前一週天氣的影響，再加上 training 和 testing data 其在時間上是連續的，所以我們把 feature 平移一週，如圖(二)所示，來做預測。

weekofyear	week_start_ndvi_ne	weekofyear	total_cases
18	30/04/1990	18	4
19	07/05/1990	19	5
20	14/05/1990	20	4
21	21/05/1990	21	3
22	28/05/1990	22	6
23	04/06/1990	23	2
24	11/06/1990	24	4
25	18/06/1990	25	5
26	25/06/1990	26	10

→

weekofyear	week_start_ndvi_ne	weekofyear	total_cases
18	30/04/1990	18	4
19	07/05/1990	19	5
20	14/05/1990	20	4
21	21/05/1990	21	3
22	28/05/1990	22	6
23	04/06/1990	23	2
24	11/06/1990	24	4
25	18/06/1990	25	5
26	25/06/1990	26	10

圖(二)

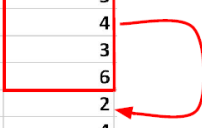
(d) 增加一個 feature

因為在不同季節感染人數應該會有差異，尤其是夏天和春天的氣候差異應該很大，再加加上一一年之中第 1 週和第 52 週天氣應該很接近，所以我們把 week\_of\_year 處理加入， $\cos\left(\left(\frac{\text{week\_of\_year}}{52}\right) * 2 * \pi\right)$ ，這樣可以使得夏天時cos 值趨近於 1，冬天時cos 值趨近於 -1，把夏天和冬天分開，並且把第 1 週和第 52 週拉近。

(e) 用 label來預測 label

當疫情爆發時，感染人數應該會節節上升，但當疫情受到控制之後感染人數應該也要漸漸減少，由此可知前幾週的感染人數應該也會影響到當週的感染人數，因此我們把前五週的 label當作 feature 來預測，如圖(三)所示，來預測當週的 label，在 training 時這很容易做到，但在 testing 時，我們是運用我們預測的數值來預測當週的數值。

weekofyear	total_cases
18	4
19	5
20	4
21	3
22	6
23	2
24	4
25	5
26	10



圖(三)

## 2. Model Description

(a) Random Forest (以下簡稱 RF)

RF 是將 ensemble method 的概念套用到 decision tree 上，可用於分類問題或迴歸問題，而在這次的題目中，我們將 RF 用於迴歸分析。首先，我們會先介紹 decision tree 如何用於迴歸分析，再來，我們會說明如何將多個 decision trees 的預測結果結合起來，形成最後的 RF 演算法。

對於一個 decision tree，第一個要決定的，即是在每一個 node，它要使用多少個 features (稱為  $m$ ) 來做決策。決定好後，我們會開始利用 training data 來 grow decision tree，在每一個 node，會從 training data 有的所有 features 中，隨機選取  $m$  個 features 作為這個 node 分裂的依據，分裂的方法是基於怎樣分才能使得被分在一起的 data 的同質性更高，衡量 data 的同質性有很多方法，如 information gain、gini index 等。那麼，關鍵的是這個 tree 要長到什麼地步呢？方法有很多種，最簡單的即是設定一個最 tree 的最大深度，除此之外，還可以額外設定一些停止成長的規範，例如: information gain 低於一定的數值。最後，每一個 leaf node 會代表一個預測數值，這個數值即為被分類到該 leaf node 的所有 training data 的 label 之平均。一般用 decision tree 做 regression 時，會透過剪枝來避免 overfitting，但在 RF 中，我們省略這套程序，用 ensemble methods 來解決 overfitting 的問題。

Random Forest，顧名思義，即是隨機的產生很多棵 decision trees 來共同做決策，在迴歸分析的例子，RF 即是將所有的 decision trees 預測的數值加總取平均，以得到此演算法最後的輸出。特別的地方是，產生每一個 decision tree 的方法，是藉由對給定的 training

data 重複的 sample 來產生該 tree 自己的 training data，此種方法稱為 bootstrap，使用 bootstrap 來 sample，每一個 tree 就無法看到所有的 training data，這即是 RF 可以解決 overfitting 的關鍵。

以下列出我們得到的最佳 RF 之模型參數中比較重要的幾個，即參數的意義：

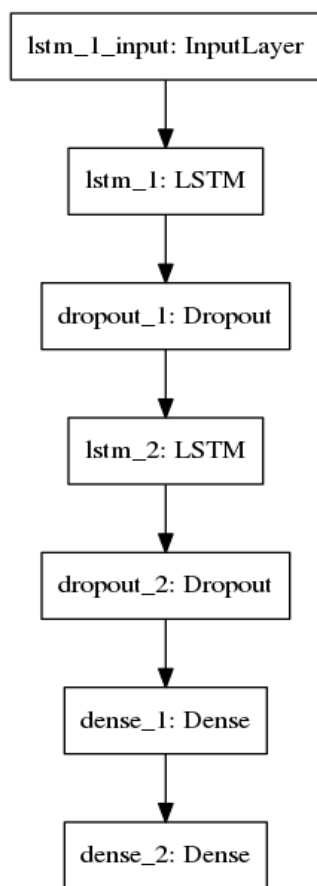
n\_estimators: 1500，代表 decision trees 的總數

max\_features: 10，代表每一個 node 參考的 features 的個數

max\_depth: 20，代表 decision trees 的最大深度

在前面 Preprocessing/Feature Engineering 的部份，我們提到過會使用前幾周的 label 作為 features。在 training 的時候，這並不會造成任何問題，因為每個時間點的 label 都是給定的，但在 predict testing data 的時候，由於所有時間點的 label 都是未知的，因此我們無法一次性的 predict 每個時間點的 label，反之，我們只能一次 predict 一個時間點的 label，並立刻將 predict 出來的結果作為接下來的幾個時間點的 feature。這種 predict 方式，稱為 One-step ahead prediction，而此種 prediction 一個顯而易見的缺點即是 error propagation：前一刻預測不精確的 label，會作為下一刻的 feature，而使得下一刻在 predict 時會用到不精確的 feature，因此這個 error 會一直傳遞下去，使得愈後面時間點的預測結果愈不精確。

#### (b) LSTM



因為這個題目的 data 是時間序列資料，因此非常自然的，會想到要試著使用 LSTM 來處理這個題目。在 RF 的 model 中，為了善用時間序列資料的特性，我們將前幾周的 label 作為 features，但在 LSTM 的架構中，因為前幾周 features 的特性會被 LSTM 記住，因此理論上我們不需要包含任何的 lagging features，而是讓 LSTM 自己去幫我完成等效的工作。這樣的架構一個立刻凸顯出來的好處就是它避免了前面提到過的在預測上 error propagation 的問題。

如左圖所示，此 network 的架構為兩層 LSTM 連接兩層 Dense layer，兩個 LSTM 的 dimension 依序為 16、8，兩個 Dense layer 的 dimension 則依序為 4、1。兩個 LSTM 都設有 0.5 的 Dropout，以用來防止 model 去 overfit 訓練資料。

在訓練的過程中，使用的 optimizer 是 Adam，loss function 為 mean absolute error。此外，DengAI 題目中的兩個不同地區對應到的 network 架構一致，但是使用的 epochs 數量不同：San Juan: 20 epochs、Puerto Rico: 40 epochs

關於 network 參數對預測結果的影響，會在後面的實驗中一併討論。

### 3. Experiments and Discussion

以下 (a) ~ (d) 的討論皆是基於 Random Forest

#### (a) 兩個地區各自 train 自己的 model 之好處

這次的 data 一共來自兩個地區，分別是 San Juan 和 Puerto Rico，由於登革熱的案例數很能會受到一些沒反映在給定的 features 上的地域性因素影響(如: 地區政府整治登革熱疫情所投入的人力、財力，或者是鄰近的區域是否有登革熱重災區等)，因此兩個城市各自訓練自己的 model 或許會是個比較好的選擇。但是這個作法的一個顯著缺點便是訓練資料的大幅降低，原本兩個城市若共用一個 model，則兩個城市的資料可以一起用來訓練，但將 model 區分開來就直接導致了可以用來訓練的 data 減半。

	sj_val_mae	iq_val_mae
model 合併	18.39	5.73
model 分開	18.04	5.64

由上表之結果可發現，確實兩個不同的地區各自訓練自己的 model 能降低 model 預測的誤差，雖然影響並沒有十分顯著。

#### (b) 平移 label 對 training 的幫助

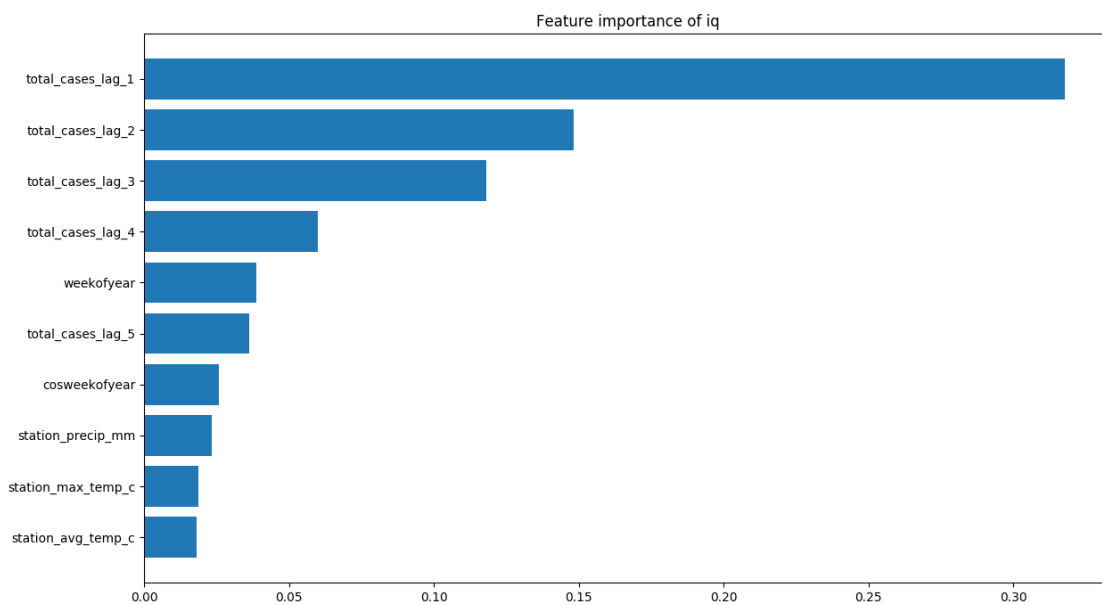
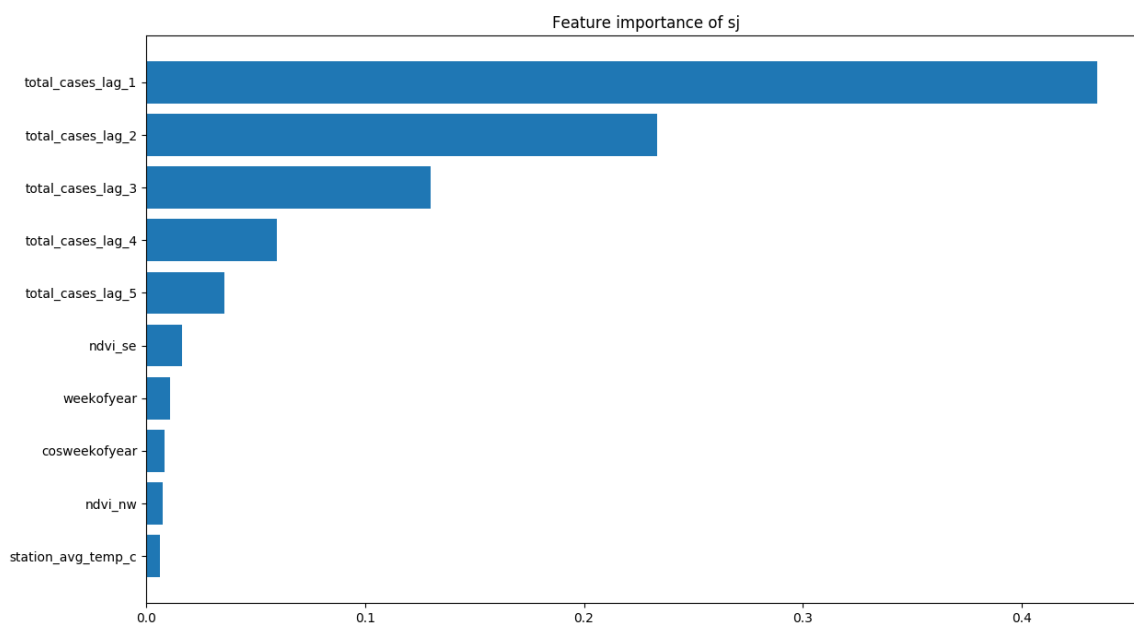
直觀的來想，當周的天氣狀況並不太可能立刻影響當周的登革熱案例數，而是應該會延遲個幾周後，影響的效果才會浮現出來。因此，在 training 的過程中，將 label 從時間上向後平移應該是個合理的作法，以下探討延遲幾周對預測結果的影響最好。

延遲週數	sj_val_mae	iq_val_mae
0	18.46	5.74
1	18.22	5.56
2	18.27	5.87

對兩個地區來說，都是將 label 往後移動一週的結果最好，因此我們可以推斷天氣狀況對登革熱的影響會有大約一周的延遲。

#### (c) 由 RF 了解各個參數的重要性

利用 sklearn 的 RF 之 feature\_importances\_ 這個 attribute，我們能得到 RF 對各個 features 的重要性之比較，以下畫出兩個不同地區的 RF 判定的前十重要的 features。



由以上兩圖可以清楚發現，無論是哪一個地區，前幾周的 label (total\_cases) 都是最重要的 feature，且其重要性遠高於其他天氣因素相關的 features。這也告訴了我們，在處理時間序列資料時，若是我們將每個時間點當作互相獨立的訓練資料來作預測，則我們會喪失掉真正有用的 features。

另外一個可以從圖表中發現的特別之處是 weekofyear 這個 features 的重要性也是高於其它的氣象要素，這代表了登革熱的案例數是以年為單位做近乎週期性的變化，因此透過

比對 weekofyear，我們可以利用過去資料中同一週的案例數來推估現在這一週的登革熱案例數。

(d) 往前看幾周的 label 最恰當

由(c)的討論可知道，與當週的 label 相關性最大的 features 為前幾週的 label 本身。因此，這裡要探討往前看起週的 label 對預測結果的影響最好。

往前看幾週的 label	sj_val_mae	iq_val_mae
0	24.26	16.31
1	18.07	10.30
2	17.50	8.93
3	17.75	6.53
4	18.13	5.94
5	18.39	5.73
6	18.52	5.75

實驗的結果如上表，在 San Juan 地區，往前看兩週的 label 之預測結果在 validation set 上之 loss 最小，而在 Puerto Rico，則是往前看五週最好。但對於兩個地區來說，完全不看之前的 label 來做預測的結果，其 loss 對兩者來說都是最大的，因此如同在 (c) 分析之結論，對於時間序列資料，使用 label 本身作為 features 是必不可少的。

(e) RF 參數實驗

改變 n\_estimators: 固定 max\_features = 10, max\_depth = 20

n_estimators	sj_val_mae	iq_val_mae
50	18.43	6.21
100	18.44	5.92
500	18.47	5.77
1000	18.37	5.80
1500	18.39	5.73

改變 max\_features: 固定 n\_estimators = 1500, max\_depth = 20

max_features	sj_val_mae	iq_val_mae
5	18.96	6.33
10	18.39	5.73
15	18.09	5.61
20	17.87	5.47
25(總共使用的 features 數)	17.76	5.40

改變 max\_depth: 固定 n\_estimators = 1500, max\_features = 25

max_depth	sj_val_mae	iq_val_mae
10	17.78	5.38
15	17.76	5.41
20	17.76	5.40
25	17.76	5.40

以上三個參數中，max\_depth 對預測結果的影響最小，可能原因是在抵達給定的 max\_depth 前，決策樹已經受到其他規範的 stopping criterion 而停止，因此 max\_depth 的效用沒有發揮出來。而 n\_estimators 在超過一定大小後(500~1000)，對於 RF 的預測之精確度的影響性也幾乎是沒有了，這個結果非常符合我們的直覺，畢竟我們不能期望透過 ensemble methods 無限制的降低 loss，因此使用了一定數量的 decision trees 後，接下來再繼續增加 n\_estimators 就只會付出多餘的計算成本，而沒辦法再降低 loss。

至於 max\_features 的實驗結果就比較有趣，在 validation set 上，隨著 max\_features 的增加，val\_loss 也跟著下降，這個現象與預想中的並不相同。導致這個現象的原因有可能是因為這次 RF 使用的總參數量太少(只有 25 個)，所以在沒個 node 都用全部的 features 去決策是一個可行的方式。

#### (f) LSTM 參數實驗

嘗試以下不同 network 架構：

input -- lstm(dim:16) -- lstm(dim: 8) -- relu(dim: 4) -- output(relu)

sj\_val\_mae: 27.73 iq\_val\_mae: 6.67

input -- lstm(dim:32) -- relu(dim: 4) -- output(relu)

sj\_val\_mae: 24.25 iq\_val\_mae: 6.32

input -- lstm(dim:16) -- lstm(dim: 8) -- output(relu)

sj\_val\_mae: 24.93 iq\_val\_mae: 6.64

比較結果，使用一層 lstm 連接一層 dense layer 的結果最好。

(g) LSTM 與 RF 之比較

根據我們的實驗結果，兩個模型在 validation set 上可達到的最好之結果分別如下：

model	sj_val_mae	iq_val_mae
RF	17.76	5.40
LSTM	24.25	6.32

實驗結果，RF 在這次的 case 之預測結果遠遠優於 LSTM。首先，因為我們實際上是投入了比較多的時間在調整 RF 上，所以也有可能 LSTM 實際上也能達到相同的結果，只是在我們有限的嘗試中無法做到。其次，這次的 training data 數量並不夠多，San Juan 地區有 937 週，Puerto Rico 地區有 520 週，neural network 的好處在於能實現一些複雜的 non-linear function，但是在 training data 如此少的情況下，overfitting 是一個嚴重的問題，這也許能稍微解釋為何 RF 和 LSTM 的結果有如此顯著的差異。

(h) validation 相關議題

這次的 final project 中，遇到最特別的問題就是 validation set 上的分數與 public leaderboard 之結果差距很大，這是在這門課的 hw1 ~ hw6 中都沒有遇到的問題。仔細討論之後，我們得出了一個結論：在使用 cross validation 時，一般的作法是直接將 training data 隨機的分成如 5 個 fold，再做 cross validation。但是這次的 data 是時間序列資料，因此在切割 validation set 時，我們不應該將 training data 隨機打散，否則可能會發生如 training set 中有第 10 週和第 12 週的資料，而第 11 週的資料在 validation set 中，由於時間資料的連續性，因此第 11 週的資料基本上和 training set 中第 10 週和第 12 週的資料會存在非常大的相似性，換句話說，這相當於我們在 training set 中看到了 validation set 中的資料，而這也就是 validation 的結果失去了價值的原因。所以，處理時間序料資料的問題時，validation set 應該是直接取前段或後段的一整個連續時間段，若是隨機選取 validation set 反而會產生上述所說的問題。