Machine Learning 2017
Final Project Report
DengAI: Predicting Desease Spread

Team Name
NTU_b02901097_大教堂時代 B

Team Members	Work Division
b02901082 杜韋頤	研究 domain knowledge、DNN、RNN
b02901093 吳岳	preprocessing、DNN、RNN
b02901095 鍾杰	random forest、ensemble、報告統整
b02901097 張哲銘	preprocessing、RNN、random forest

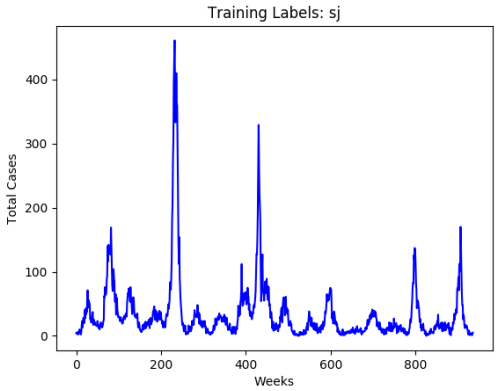
1. Problem Description

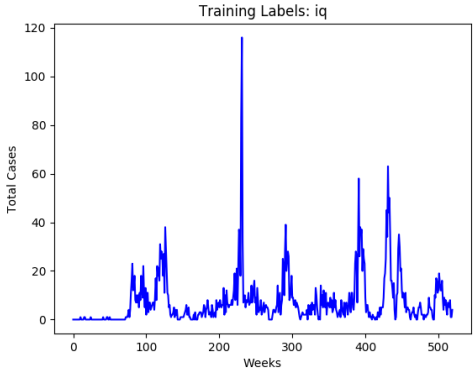
每年夏天，控制登革熱疫情都是位於熱帶與亞熱帶各國政府的首要之務。登革熱案例往往與氣候密切相關，本題希望能找出適合的模型，以氣候資訊進行案例數預測。我們針對兩個城市進行預測，分別為 San Juan – Puerto Rico 以及 Iquitos – Peru，接下來都會簡稱為 sj 與 iq。

2. Preprocessing/Feature Engineering

2.1. Training Features

這個題目要預測的城市分為兩個，各自訓練資料中案例數對時間作圖如下，可以明顯看到 sj 跟 iq 的案例數與趨勢分佈截然不同，因此首先就要將兩個城市的數據分開來各自訓練，以得到最佳效果。



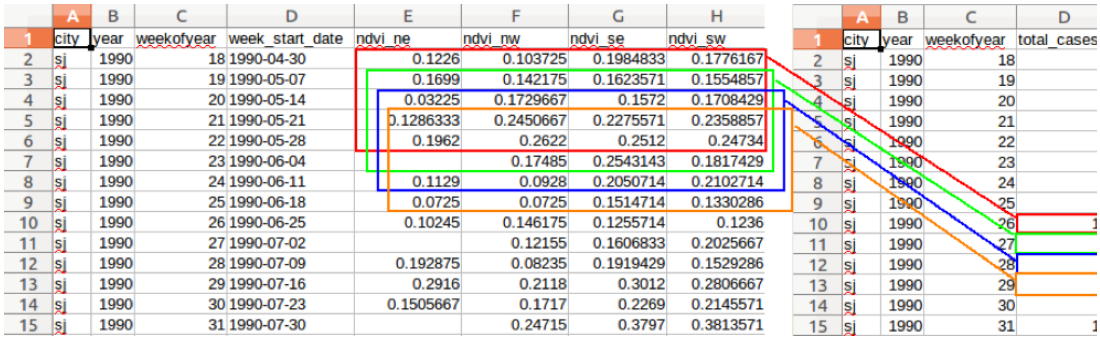


此外，此題 feature 以氣象資訊為主，而各項數據對案例實際的相關性也不同，因此我們經過嘗試選擇最適合的各個模型的數據。這部份詳細的選擇將會在各模型架構設計的地方進一步說明。

再者，蚊子有其生長週期，有理想的溫溼度條件的當週會促成病媒蚊的大量繁殖，但從卵到成蟲再到大量肆虐又會經過 5 至 10 週的潛伏期。另外，不同年份或時期的案例數也可能有各自的 bias 跟趨勢。考慮到上述原因，我們主要會對數據進行兩種處理來得到訓練資料，分別是 shift 和 lagging。

2.2. Shift and Lagging

原本的訓練資料是同一週的 label 與 feature 放在一起，shift 顧名思義就是調整兩者間的配對，我們通常會將某一週 label 對應到大概一至四週以前的 feature。此外，在處理時間連續資料時，很難真的斷定就是第幾週前的天氣狀況導致本週的案例，因此除了 shift 一定週數，還會從那週再往前多取連續幾週的數據使用，這就是 lagging。下圖是對 feature 做處理，shift 為 4、lagging 為 5 的範例。



	A	B	C	D	E	F	G	H
1	city	year	weekofyear	week_start_date	ndvi_ne	ndvi_nw	ndvi_se	ndvi_sw
2	sj	1990	18	1990-04-30	0.1226	0.103725	0.1984833	0.1776167
3	sj	1990	19	1990-05-07	0.1699	0.142175	0.1623571	0.1554857
4	sj	1990	20	1990-05-14	0.03225	0.1729667	0.1572	0.1708429
5	sj	1990	21	1990-05-21	0.1286333	0.2450667	0.2275571	0.2358857
6	sj	1990	22	1990-05-28	0.1962	0.2622	0.2512	0.24734
7	sj	1990	23	1990-06-04		0.17485	0.2543143	0.1817429
8	sj	1990	24	1990-06-11	0.1129	0.0928	0.2050714	0.2102714
9	sj	1990	25	1990-06-18	0.0725	0.0725	0.1514714	0.1330286
10	sj	1990	26	1990-06-25	0.10245	0.146175	0.1255714	0.1236
11	sj	1990	27	1990-07-02		0.12155	0.1606833	0.2025667
12	sj	1990	28	1990-07-09	0.192875	0.08235	0.1919429	0.1529286
13	sj	1990	29	1990-07-16	0.2916	0.2118	0.3012	0.2806667
14	sj	1990	30	1990-07-23	0.1505667	0.1717	0.2269	0.2145571
15	sj	1990	31	1990-07-30		0.24715	0.3797	0.3813571

	A	B	C	D
1	city	year	weekofyear	total_cases
2	sj	1990	18	
3	sj	1990	19	
4	sj	1990	20	
5	sj	1990	21	
6	sj	1990	22	
7	sj	1990	23	
8	sj	1990	24	
9	sj	1990	25	
10	sj	1990	26	1
11	sj	1990	27	
12	sj	1990	28	
13	sj	1990	29	
14	sj	1990	30	
15	sj	1990	31	1

除了對 feature 作 lagging 處理，我們也會使用在 label 上，取用前幾週的案例數加到我們的訓練資料中，這些資料比天氣資訊更能夠顯示出案例數在某段時間內的趨勢起伏。我們針對訓練資料作的 shift 或 lagging 數都會於各個模型介紹時說明。在做預測時，測試資料的 feature 如果有 shift 或 lagging 時也必須往前取。因為測試資料開頭跟訓練資料結尾的時間是相連的，因此做法就是根據需要取訓練資料最後幾週來提供給測試資料的頭幾筆資料使用。

2.3. Filling Missing Data

原始訓練資料有些地方都是有缺的，如果直接以 0 填補不合理，例如前一天最高溫有 35 度，隔天因資料有缺被補為 0，變動幅度太大。我們考慮到天氣資訊是時間連續的，因此取前面一週的數值填補，前一週同樣空缺則繼續往前取到有值的地方。這部分處理我們是使用 pandas 內建涵式 fillna(method='ffill')。

2.4. Other Techniques

除了訓練資料會做特殊處理外，我們在訓練時考慮到時間連續的資料特性，也嘗試採用不同 validation 方法以及對預測結果的調整。Validation 上我們使用 rolling cross validation，假設我們做 5-fold cross validation，則先將時間上連續的訓練資料切成六等份，再根據下表所示，選擇每一次的訓練及測試資料來計算 loss，最後平均起來作為該模型 validation 得到的 loss。此外，我們也嘗試對預測出的結果做移動平均，使連續的前後週之間案例數變化更順暢，試驗結果顯示 window size 為 3 時效果最好，loss 最多可以減少約 0.5。

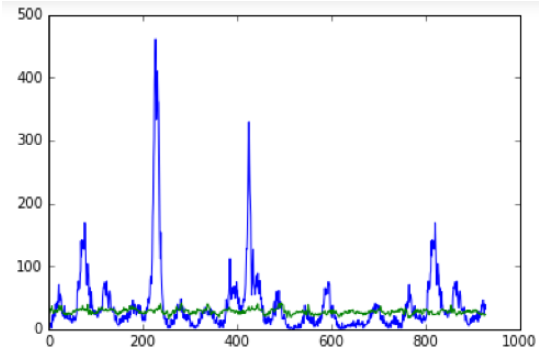
Fold	Training Set	Testing Set
1	[1]	[2]
2	[1 2]	[3]
3	[1 2 3]	[4]
4	[1 2 3 4]	[5]
5	[1 2 3 4 5]	[6]

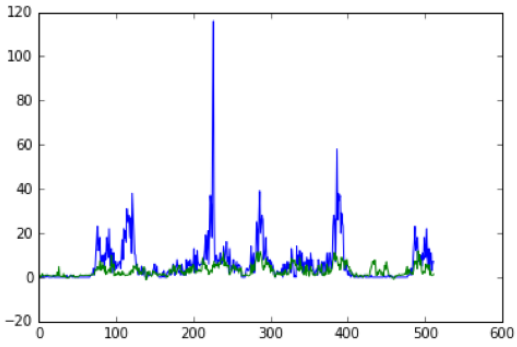
3. Model Description

3.1. DNN

我們一開始始嘗試以 DNN 預測，模型架構大概是有數層 dense layer，最後 output layer activation 為線性輸出得到一個值。首先去除數據中的年份、週數、日期與 nvdi_ne(空缺的數據太多)並以前述方式填補空缺的部份。如此處理過後，單週會有 19 項數據，所以預測一個 label 所用的 feature 數目為 19 乘上 lagging 數。

我們嘗試 shift 與 lagging 在 1 至 10 之間的各種組合，以及將 DNN 加深加廣，但預測出來的結果都無法超越 Simple Baseline，最好只在 26 到 28 之間。 iq 在訓練資料上的預測結果在 label 小時尚可行，但無法預測出極端的高峰值(左下圖)；而在 sj 預測結果則近乎水平線，根本沒有無法 fit 到訓練資料(右下圖)，造成誤差極大。下圖中藍線為原始 label，綠線為 DNN 預測結果。





3.2. Random Forest

我們嘗試的第二種模型為 random forest，因為是 regression 模型我們以 sklearn 的 RandomForestRegressor 實作。目前訓練出來最好的一個模型前處理及參數設定如下：氣候指標選用與案例數相關係數最高的 reanalysis_dew_point_temp_k、reanalysis_specific_humidity_g_per_kg、station_min_temp_c 和 station_avg_temp_c，再加上 week。至於 shift 為 0，lagging 部分則為 10，換句話說是利用當週與往前 9 週的 feature 來預測該週的結果。兩個城市訓練模型所設定的參數如下表。

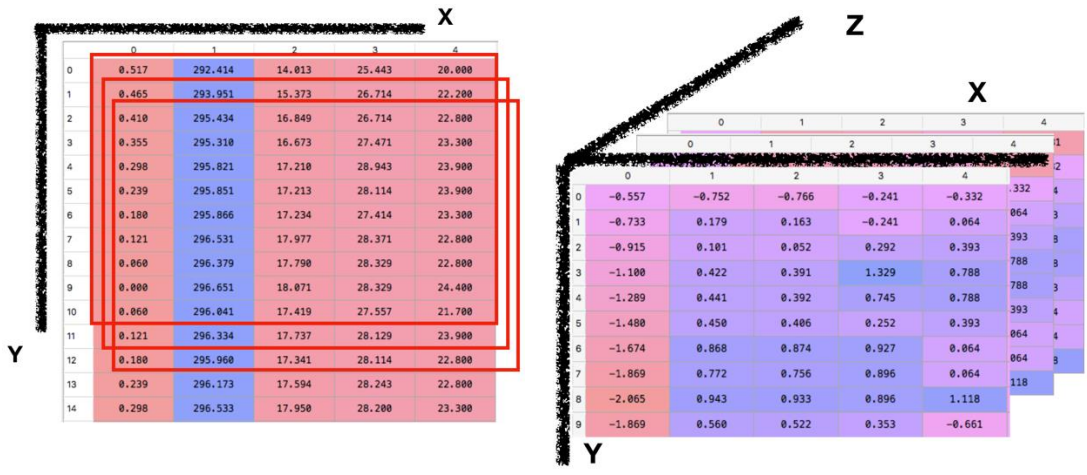
City	max_depth	max_features	min_samples_leaf	min_samples_split	n_estimators
sj	19	1	2	2	383
iq	1	47	71	89	436

我們後來又加入 lagging label 來訓練 random forest 模型，但因嘗試時間較短，並未能訓練出較佳的成績。目前這個模型可以跑出最好的結果是採用上面提到的四項相關度最高的指標及各自的三次方，再加入 label 後以 shift 為 1、lagging 為 10 來擴增，因此每筆用於預測一週案例數的資料共有 90 項，選出來的各項 feature 會接著進行標準化後才拿來訓練。

3.3. RNN

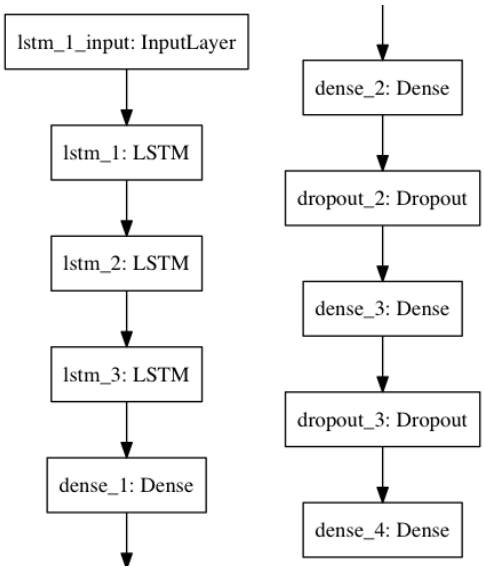
基於作業五的經驗，我們知道 RNN 可以用來學習前後連續的字詞，既然我們有時間連續的資料，且想判斷彼此間的關係，為何不將數據轉成一個一個字詞用 RNN 預測看看呢？然而，輸入給 RNN 的訓練資料必須經特殊處理才符合需求。

首先，我們必須讓原本二維的 training set 變成三維，如下圖所示，左邊是原始的 feature，右下圖是修改過要放入 RNN 的訓練資料。假設要取連續 10 週的資料，那我們就會以左下圖紅色框框為一個 set，按第 1 到 10 週、2 到 11 週等以此類推依序取值，並對每個 set 做標準化。最後，再把這些 set 在 Z 軸依序並排成一個三維陣列，放入 RNN。



我們也嘗試了很多 LSTM 的架構，右表分別是固定的參數跟嘗試改變的參數，左下圖則是在三層 dense layer 加上三層 LSTM layer 下的架構圖。

固定參數	數值
batch size	32
optimizer	Adam
RNN activation	tanh
dense activation	relu
output activation	sigmoid
變化參數	嘗試數值
# of dense layer	1~3
# of RNN layer	1~3
dropout	0.4~0.7
# of lagging (週數)	5, 8~12



根據我們調過不同的參數顯示，連續預測的週數只要介於 5 到 12 之間，最好模型的 loss 基本上都不會差太多。即便有幾組參數，如 dense layer 數為 1、LSTM 為 2 時的 validation loss 能比其他組合低，但上傳後都無顯著的差異。在這樣的狀況下，我們決定取數個不同架構或參數組合的模型做 ensemble，有多組成功突破 Strong Baseline，最好的得到了 22.21，其架構設計跟所使用的 feature 如下表。

模型	feature 有無加上平方項	# of dense layer	# of RNN layer	# of lagging
1	無	1	1	10
2	有	2	2	10
3	有	2	2	12
4	無	2	3	10
5	無(改用週數取 cosine)	2	3	5

4. Experiments and Discussion

4.1 Training Random Forest without Lagging Label

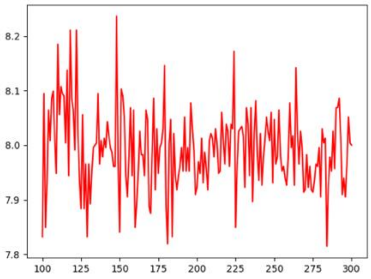
在訓練這種模型時，我們使用了 hyperopt 這個 package 來快速大量嘗試不同的 RandomForestRegressor 設定參數的組合，調整的參數有 max_depth、max_features、min_samples_leaf、min_samples_split 與 n_estimators 共五項。hyperopt 會根據設定重複若干迴圈，每一次會隨機從給定區間取樣作為此次生成 RandomForestRegressor 模型的設定參數，相當方便。

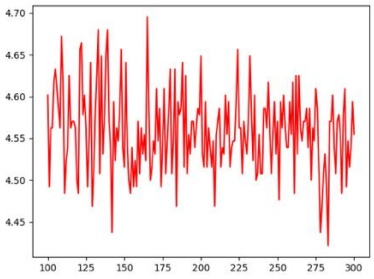
不過根據不同參數嘗試的結果發現，其實每次以不同前處理完的訓練資料做訓練，得到的各個參數彼此間並沒有太一致的規則，例如 n_estinmator 都會介於 300 到 350 之類的，反而是變動很大，就算是相同前處理的訓練資料找到的參數也會略有不同。

另外，根據我們試出來的結果，單用 random forest 只要參數找得對、好好思考一下應該使用哪些較具代表性的 feature，在不加入 lagging label 的情況下(但需要 lagging feature)就可以勉強掠過 Strong Baseline。

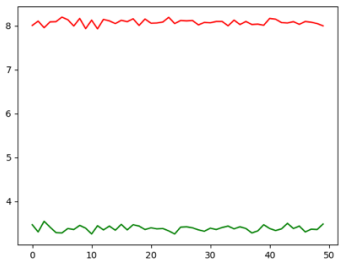
4.2. Training Random Forest with Lagging Label

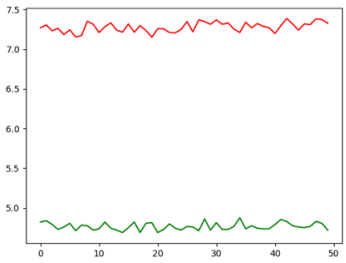
針對這個模型，我們做了以下兩種實驗：調整 sj 和 iq city 合併預測的 tree 的數目以及加入 feature 的二次方或三次方項。第一種實驗針對 sj 和 iq 兩個城市不同 tree 的數目得到下圖的結果，橫軸是 tree 的數目，縱軸是 validation loss (此處是用一般的 k-fold cross validation，引此數值皆偏小)。我們發現其實不管對於 sj (左下圖)或是 iq (右下圖)來說，tree 的數目對於 loss 的影響不大，因此我的作法是選擇 loss 變異數相對小的區間來確保結果的一致性，因此 sj 就是大概在 250 個 tree，而 iq 就是大概在 150 個 tree 的地方。





第二種實驗是調整所選擇的 4 種 feature 在要新增的次方項，下圖為用實驗次數當橫軸，validation loss 當縱軸的作圖。紅線是代表 sj 的 validation loss，綠線則是 iq 的；左下與右下圖分別為加入二次及三次項。可以發現用三次方項當作 feature 和用二次方項當作 feature 的差別在於，平均來說 sj 的 loss 在使用三次方 feature 預測時是比較穩定的介於 7 到 7.5 之間，但使用二次方時卻會在 8 左右變動。考量到 sj 的預測準確度對於成績的影響較大，所以我們選擇以三次方當作新增的 feature。





4.3. RNN

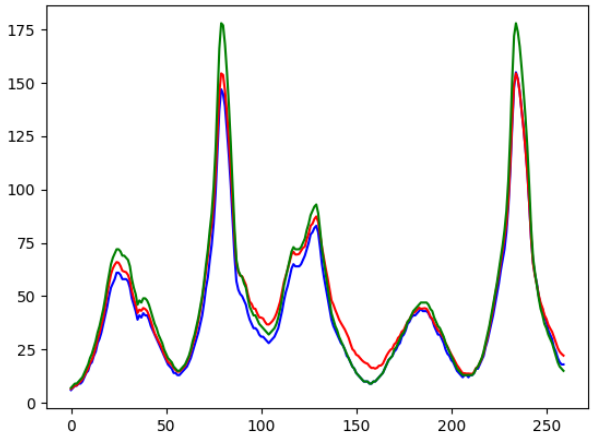
現在比較 DNN 與 RNN，兩個都有設定 Early Stopping 跟 Model Check Point 的機制。DNN loss 最小的模型通常都出現在前 10 個 epoch，而 RNN loss 最小的模型平均落在第 50 至 60 個 epoch 左右，訓練途中最佳模型的更新次數明顯比 DNN 多，自然也比 DNN 訓練更久，在這樣的結果下，DNN 在 sj 的 validation loss 大概落在 20 到 23 之間，而 RNN 的 loss 可落在 15 到 17。看到 RNN 的訓練成效比 DNN 好很多，原本預期上傳後也可以突飛猛進，卻仍落在 25.5 到 27 之間。RNN 成效是比 DNN 好，但這點改善還不夠讓我們破 Strong Baseline。

不過在此同時我們也想到該加入 lagging label 一起做預測，以此方法得到的訓練資料會要訓練更久，但成效都比之前的 RNN 好。sj 的 validation loss 可降到 9 到 10 左右，iq 的 loss 甚至可以小於 1，至於上傳的成績落在 24 到 25 左右。如前面 RNN 模型介紹所說，最後我們再採用多個 RNN 的 ensemble 可以將成績提升至 22 左右。

4.4. Ensemble Different Types of Models

我們訓練了若干種模型，如 RNN 部分所述，各自做 ensemble 最好也就只有 22 到 23 左右，為了更上一層樓，我們決定混合不同種類模型的預測結果，並藉由觀察各自預測結果的優劣選擇合併的方式。我們首先嘗試使用 RNN 跟有加 lagging label 的 random forest 模型。

可以先觀察下圖，縱軸是 sj 在測試資料上的預測案例數，橫軸則是週數，綠線是單獨利用 RNN 上傳最好的結果；藍線是 RNN 和有加 lagging label 的 random forest 模型以 8 比 2 的比例 ensemble 的結果；紅線則是 7 比 3。我們還有試過其它比例，不過為了方便作圖只取這三種來比較。

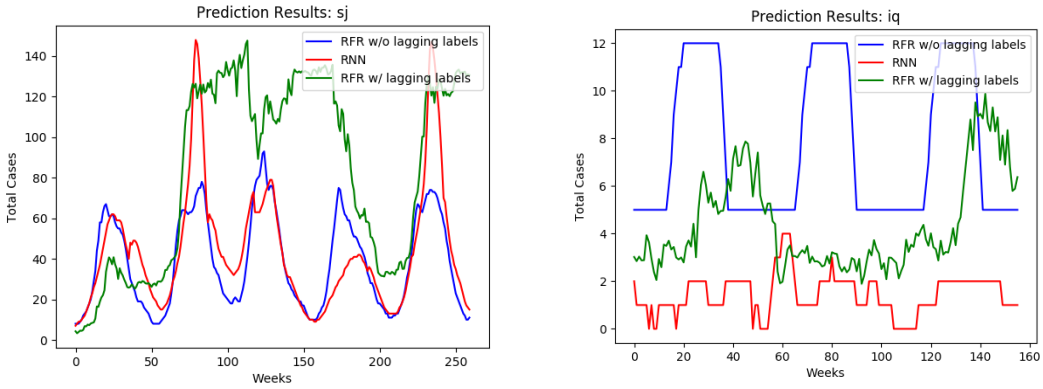


從圖中可以觀察到，單獨利用 RNN 來預測的問題在於峰值的部分會衝的過高，導致整體的 loss 會稍微增加，然而它的優點是在其餘的部分相對準確；接

著我們觀察藍線，其在峰值的地方在跟 RNN ensemble 之後做了一些修正，使其略為降低，在其他的地方和單純 RNN 的預測結果相當接近；對比紅線在峰值得地方衝的稍微高一點，不過仍然有修正到單獨 RNN 的缺失，可是在其他部分卻和 RNN 的預測結果有點差距，這導致在上傳之後成績不理想。因此在綜合觀察評判後，我們選擇以 8 比 2 的比例結合這兩種模型的預測結果。這個 ensemble 的結果可以得到 21 多接近 22 的成績。

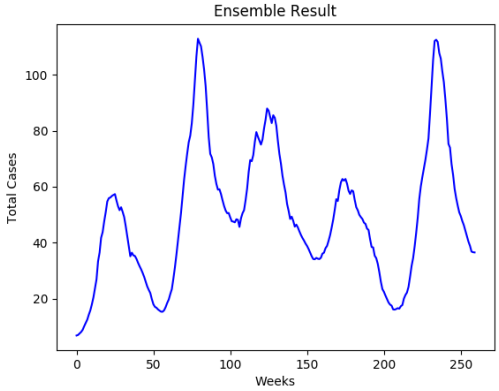
4.5. Deciding Best Model

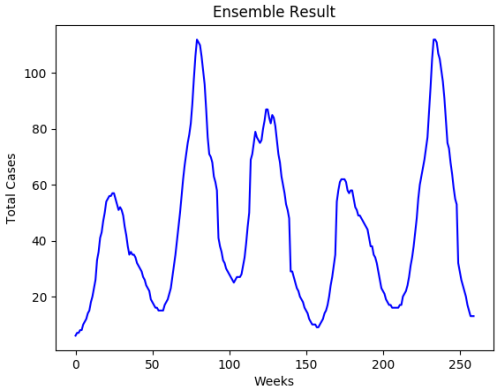
接著我們決定採用有加跟沒加 lagging label 的兩種 random forest 模型最好的結果來搭配最好的 RNN ensemble 來合併預測。左下及右下圖分別為三個模型各自預測 sj 跟 iq 的結果，其中藍線是沒有使用 lagging label 的 random forest，紅線為 RNN，綠線則是有使用 lagging label 的 random forest，下面討論中將會分別簡稱為藍線、紅線與綠線模型。



首先就 sj 來看，藍線模型以年為週期會有規律起伏，但一般峰值最多只有到接近 100 左右。觀察訓練用 label (第一張圖)可以發現某些年可能疫情特別嚴重，登革熱案例會暴增至 200、300 以上，這方面紅線模型比較接近這樣狀況，某一年可能疫情較和緩，某些年卻會有極大峰值。同樣的，綠線模型因為有加入 lagging label 來訓練，較能預測出突然爆發的峰值，但整體數值明顯偏高看不出季節的週期性。

因此，綜合三個模型預測的特點，合併時以藍線及紅線模型權重較高，綠線模型權重較低做為峰值計算的輔助項。此外，我們選擇的綠線模型並沒有訓練得很好，雖然可以出現較大峰值，但一升上去就降不下來了，會連帶影響整年週期性的起伏。因此我們改成照一週一週的結果看，若綠線模型剪掉藍線與紅線模型的平均超過一定值，就只用藍線與綠線模型的綜合結果，合併出來的結果如左下圖所示，可看見峰值都差不多但中間的幾個峰谷明顯較左圖低且穩定。

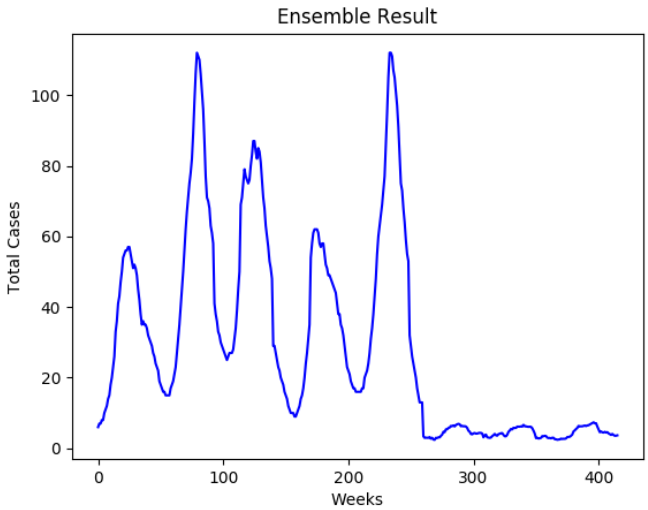




接著看 iq，藍線模型的值比另外兩個稍大一些，且有較明顯季節的週期性；綠線與紅線則數值偏小，但綠線模型比較有突然出現峰值的可能。實際 iq 的訓練 label 週期性不是特別明顯，但偶而仍有突然出現的峰值。

我們經過嘗試選擇出來的兩個城市各自組合比例如下表，而整體預測結果見下圖。兩個城市的預測結果分別以不同方式合併以後可得到最終預測結果，這個組合也讓我們上傳後得到 21.5745，是我們目前最佳的成績。

City	綠線減紅藍線平均	藍線模型	紅線模型	綠線模型
sj	>=90	0.5	0.5	0
	<90	0.4	0.4	0.2
iq	-	0.4	0.4	0.2



5. References

[1] DengAI: Predicting Disease Spread - Benchmark
http://blog.drivendata.org/2016/12/23/dengue-benchmark/

[2] Regression Tutorial with the Keras Deep Learning Library in Python
http://machinelearningmastery.com/regression-tutorial-keras-deep-learning-library-python/

[3] Scikit Learn 1.1.Generalized Linear Models
http://scikit-learn.org/stable/modules/linear_model.html

[4] Using k-fold cross-validation for time-series model selection
https://stats.stackexchange.com/questions/14099/using-k-fold-cross-validation-for-time-series-model-selection

[5] Moving Average Smoothing for Data Preparation, Feature Engineering, and Time Series Forecasting with Python
http://machinelearningmastery.com/moving-average-smoothing-for-time-series-forecasting-python/

[6] A comprehensive beginner's guide to create a Time Series Forecast (with Codes in Python)
https://www.analyticsvidhya.com/blog/2016/02/time-series-forecasting-codes-python/