ML Final Project

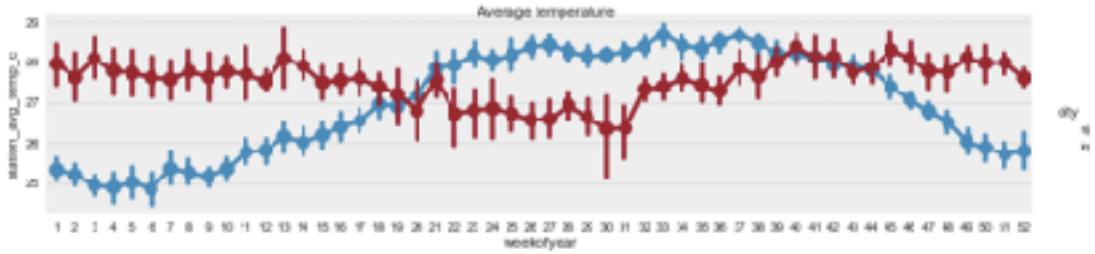
B03705012_CPUTeam

B03705006 侯舜元：程式、文件撰寫
B03705012 張晉華：程式、文件撰寫
B03705020 周祐鈞：文件撰寫、整理
B03705037 陳主牧：程式、文件撰寫

Data Preprocessing

我們猜測登革熱罹患率高的地方會是炎熱、且降雨頻繁的地區，所以我們先對「氣溫、降雨率」做分析：

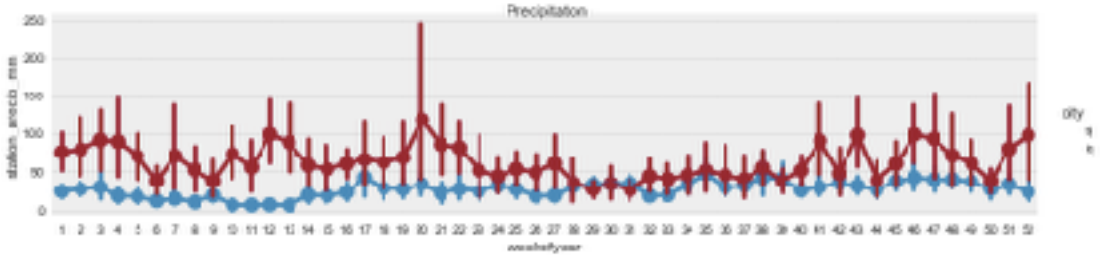
下圖為兩個城市的平均氣溫比較，藍色為San Juan而紅色為Iquitos，可以發現Iquitos的氣溫一直都維持在27,28度左右；而San Juan氣溫變化較大，冬天時約25,26度，夏天則會上升到28,29度。



經過計算求出Iquitos的平均氣溫較高。

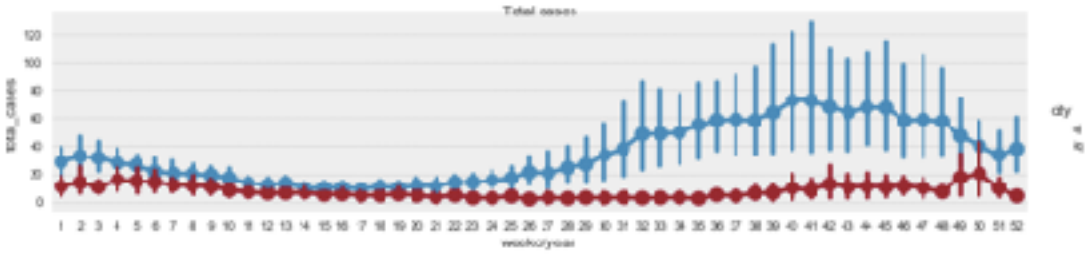
	Sj	Iq
station_avg_temp_c	27.006528	27.530933

下圖為兩個城市的平均降雨量比較，藍色為San Juan而紅色為Iquitos，可以發現Iquitos整年的降雨量幾乎都比San Juan高出許多。



	Sj	Iq
station_precip_mm	26.785484	62.467262

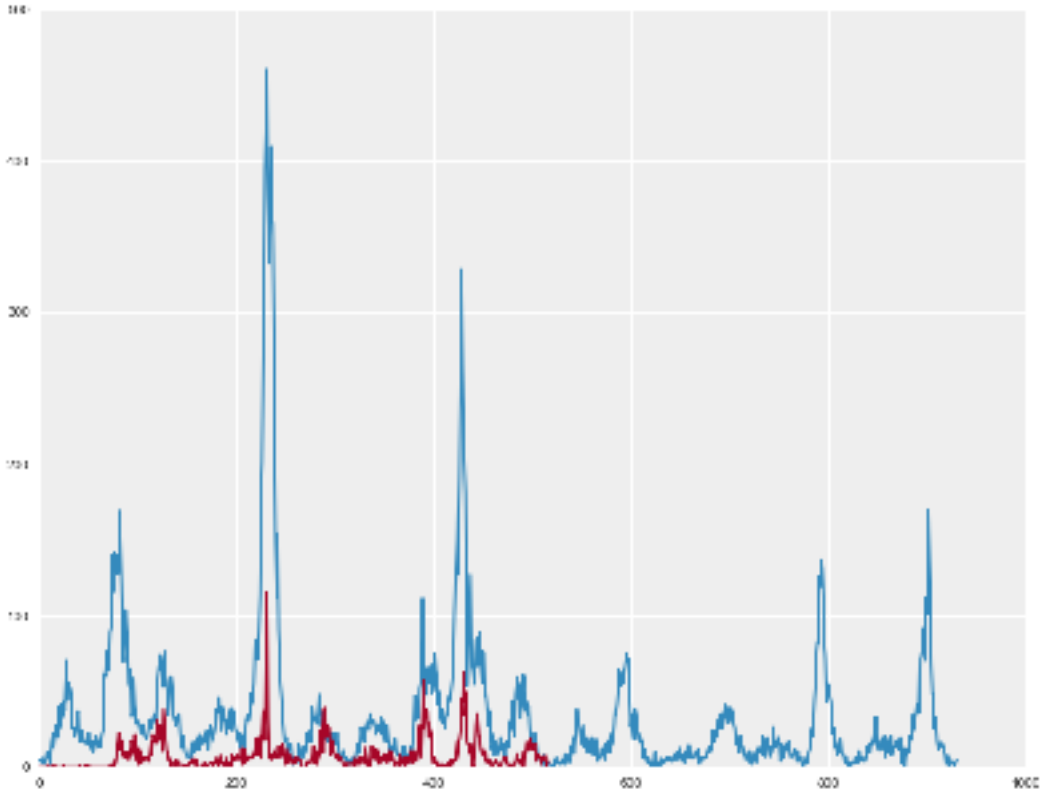
根據以上兩個因素的比較，我們推測氣溫較高且降雨量較多的Iquitos罹患登革熱的人數應該較多，但實際上的結果卻並非如此。
下圖為兩個城市罹患登革熱的人數比較，藍色為San Juan而紅色為Iquitos，可以看出San Juan罹患登革熱的人數明顯較多。



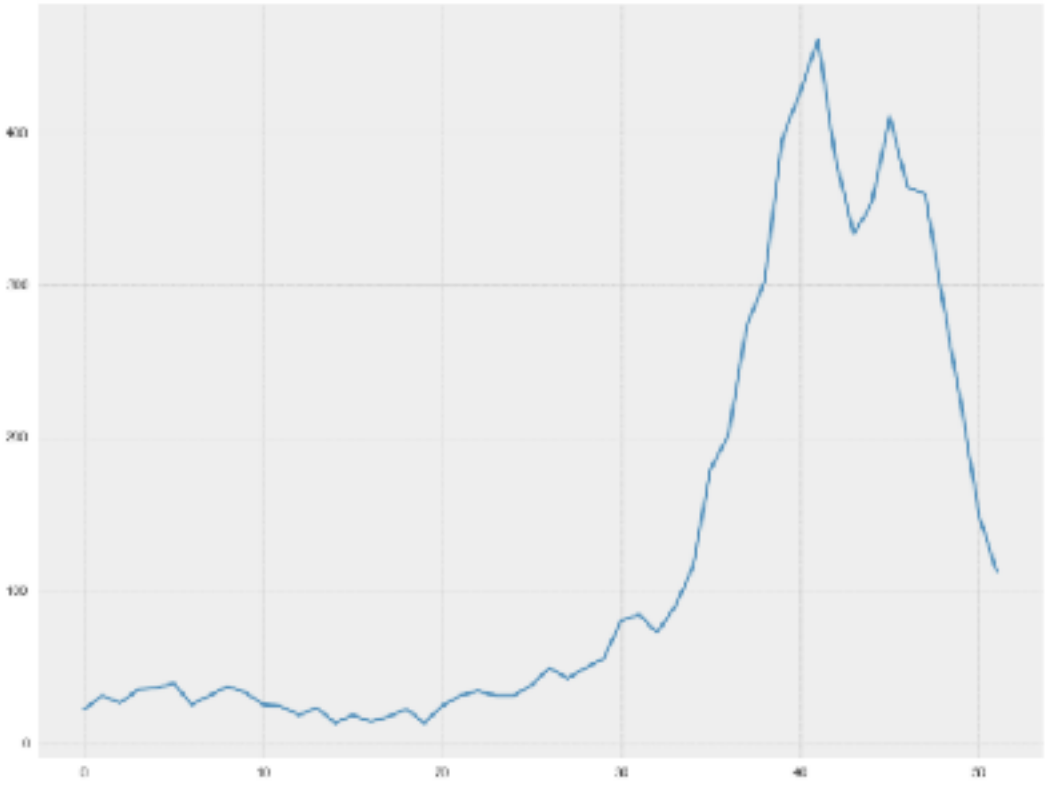
	Sj	Iq
total_cases	34.122581	7.596899

我們接著開始觀察罹患登革熱的人數在時間上的變化：
下圖為登革熱的罹患人數在時間上的變化，藍色為San Juan而紅色為Iquitos，可以發現San Juan在下半年的罹患人數相對於上半年增加不少，Iquitos則是在年底有較明顯的變化。

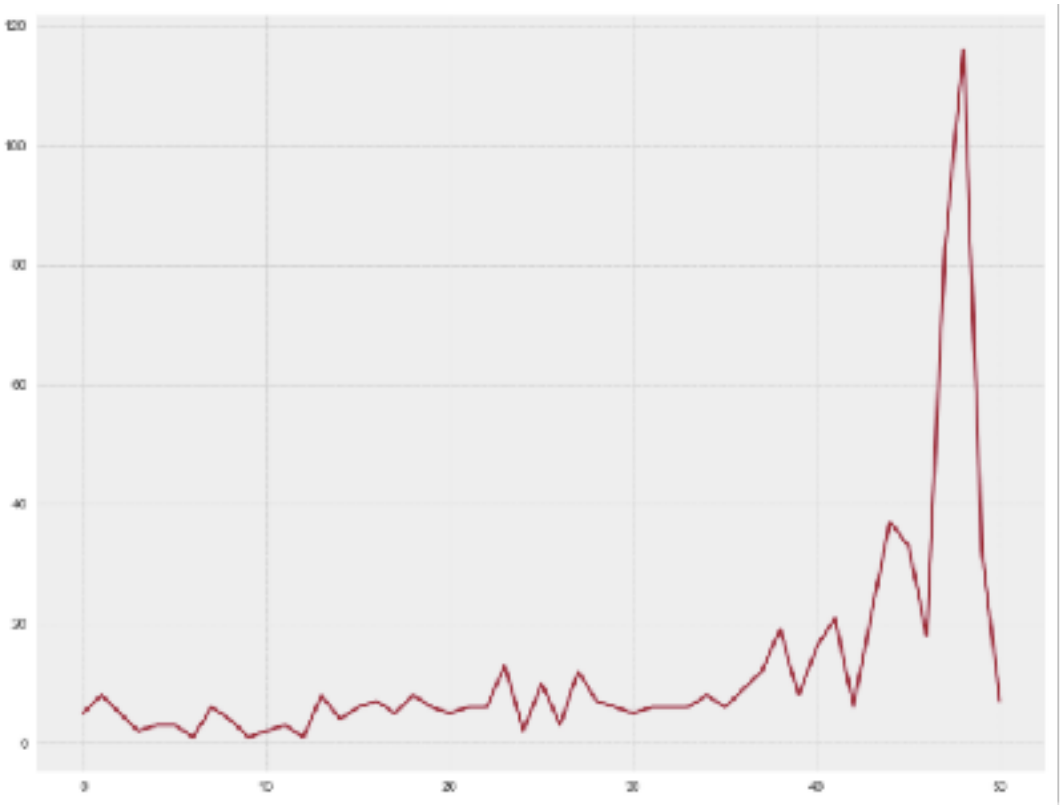
下圖為1990~2007年登革熱在兩個城市的罹患人數變化



下圖為San Juan於1994年的登革熱罹患人數變化，也是於下半年開始人數變多，且在約week 39到達一個高峰



下圖為Iquitos於2004年的登革熱罹患人數變化，與San Juan相比人數少了許多，蛋在年底時人數突然激增到了100左右



Model Description

我們選用的feature為 「year」、「weekofyear」、「ndvi_nw」、「ndvi_se」、「ndvi_sw」、「precipitation_amt_mm」、「reanalysis_air_temp_k」、「reanalysis_avg_temp_k」、「reanalysis_dew_point_temp_k」、「reanalysis_max_air_temp_k」、「reanalysis_min_air_temp_k」、「reanalysis_precip_amt_kg_per_m2」、「reanalysis_relative_humidity_percent」、「reanalysis_sat_precip_amt_mm」、「reanalysis_specific_humidity_g_per_kg」、「reanalysis_tdtr_k」、「station_avg_temp_c」、「station_diur_temp_c」、「station_max_temp_c」、「station_min_temp_c」、「station_precip_mm」、「total_case_by_month」共22個，其中「total_case_by_month」為某月份五年來的平均totalcase數，並且嘗試使用以下四個Model做出了不同的結果。

(1)DNN

以上面的feature來train，結果卡在Local min，訓練不起來。推測可能原因是參數過多、單位差距過大 (如濕度、座標等，經簡單的標準化之後其物理意義與數值分布差異仍大)，而我們並未進一步找出適當的正規化方法。

(2)RNN

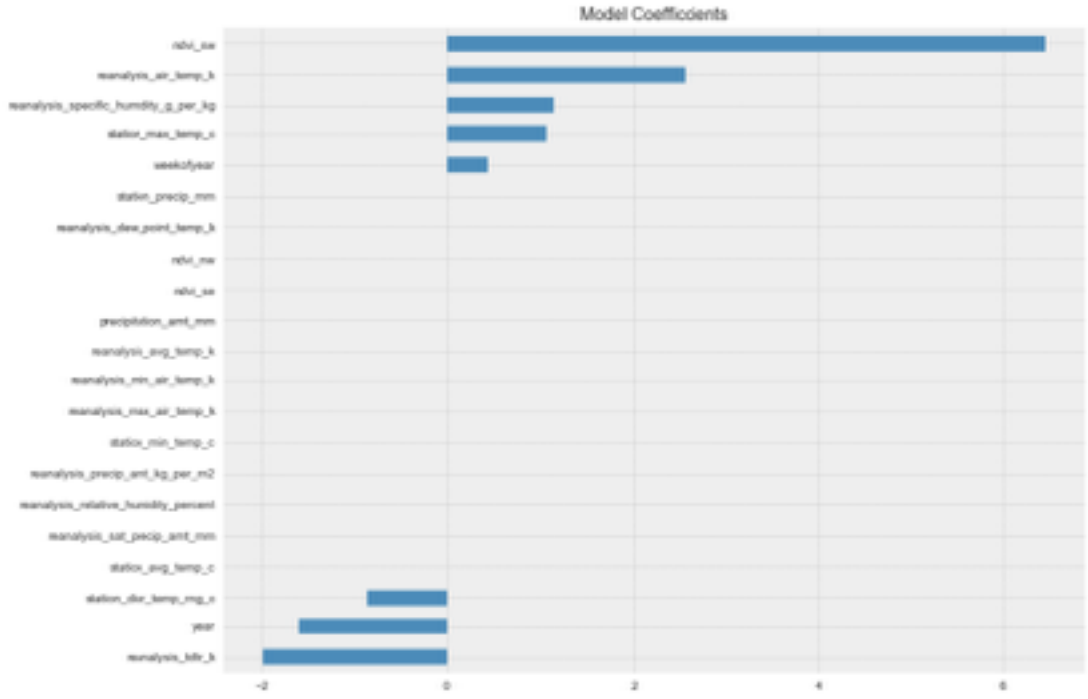
與DNN面臨同樣問題，結果亦為失敗。

(3)Random Forest

當初嘗試了Lasso regression與Random Forest regression兩者做比較，使用的feature為以上21個，得出了以下的結果：

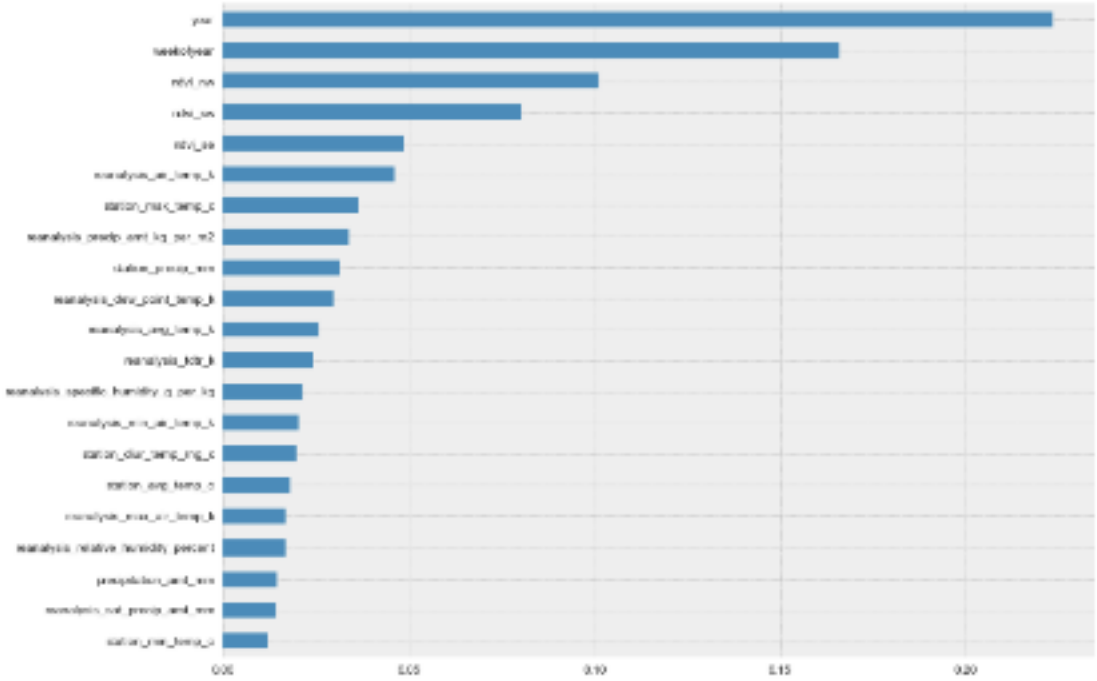
	Lasso	Random Forest
MAE	20.02	4.569
Mean	21.24	20.06
Median	19.39	17.81
Std	10.71	12.61

下圖為Lasso Regression train出的參數經過排序後畫出的圖，較有影響力的參數為「ndvi_sw」、「reanalysis_air_temp_k」、「reanalysis_tdtr_k」、「year」



下圖為Random Forest Regression train出的參數經過排序後畫出的圖，較有影

響力的參數為「year」、「weekofyear」、「ndvi_nw」、「ndvi_sw」，可以發現時間造成的影響相對較大。



比較Lasso及Random Forest所得到的結果之後，發現Random Forest的表現比較好，因此嘗試使用此model上傳，得到的分數為26.3173

Improvement

為了得到更好的結果，我們繼續使用Random Forest Regression，並將兩個城市分開來做。

以下為使用Grid Search所試出表現最好的參數：

	Sj	iq
max_features	12	10
min_samples_split	88	19
n_estimators	674	200
max_depth	18	16
min_samples_leaf	3	5

(Grid Search: 1 ≦ max_features ≦ 21
2 ≦ min_samples_split ≦ 100
100 ≦ n_estimators ≦ 1000
1 ≦ max_depth ≦ 32
1 ≦ min_samples_leaf ≦ 10)

以此model上傳出的結果為 24.9688

(4)XGBoost (XGClassifier)

從使用的參數可以看出，XGBoostClassifier與之後使用到的XGBoost主要差異是其進行binary classification。在此模型中，我們使用與random forest 一樣的feature，參數如下：

	Sj	iq
n_estimate	542	242
max_depth	6	4
colsample_byvalue	0.7	0.7
objective	binary:logistic	binary:logistic
booster	gbtree	gbtree

因得到的分數不及Random forest (最好僅達leader board 25.x)，此model後來被暫時棄用。

XGBoost: first attempt

聽完發表會後，我們發現相當多組別皆使用我們發現相當多組別皆使用XGBo

eta	0.2
subsample	0.7
objective	reg:linear
num_round	400
num_model	1
test size	0.1

以及歷

- **Model Comparison**

  我們一開始用XGBClassifier(Implementation of the scikit-learn API)跟XGBoost做比較，兩者上傳結果MAE分別為：

  XGBClassifier：28.149

  XGBoost：25.6611

猜測XGBClassifier本身會忽略掉數值連續性的成長關係，因此我們以表現較好的XGBoost為Model進行Improvement。故之後便皆改用純正的XGBoost。

eta	0.2
subsample	0.7
objective	reg:linear
num_round	400
num_model	200
test size	0.05

XGBoost: strong baseline

後來我們對XGBoost進行了更多修改，茲列如下：

- Some practical improvement

1. Feature Bagging + Ensemble：

我們嘗試使用多組XGBoost Model來預測結果，每次sample出一半的

eta	0.02
subsample	0.7
objective	reg:linear
num_round	400
num_model	1
test size	0.05

age

數量

2. 增加前四週的天氣feature

由於上面測試Ensemble時發現天氣對趨勢的影響，我們嘗試將前後後周的天氣feature一起加入(沒有資料的設為中位數)當作feature。有嘗試四

周、八週、十二周等，最後結果是四周最佳。另外我們發現單一model即可有不錯的效果，上傳結果MAE = 20.6202

eta	0.02
subsample	0.7
objective	reg:linear
num_round	400
num_model	1
test size	0.01

- Parameter Engineering

最後的優化是在針對參數，這造成了相當顯著的進步。

1. Tune xgboost的參數，將n_estimate改為預設。另外，xgboost在eval_metric的設定上，我們沒注意到document上說的

   User can add multiple evaluation metrics, for python user, remember to pass the metrics in as list of parameters pairs instead of map, so that latter 'eval_metric' won't override previous one

所以我們驚覺原本的eval_metric是用rmse，馬上改為mae

2. 將沒有資料(NaN)Dropout的threshold調低，test size切小，換不同的seed，用一樣的model不斷去嘗試，最後成功train出leaderboard18.4的結果

Appendix：Reproduce 方法

data放在final/data的路徑下
在final資料夾下執行 ./reproduce.sh

Appendix：Reference

https://github.com/du-phan/DengAI