# ML Final Project Report

組名：NTU_b02902069_CableNewsNetwor
組員：B02902044 陳映紅、B02902069 王皓正、B02902094 葉光哲

✧ 題目：DengAI: Predicting Disease Spread

✧ 分工

陳映紅：XGBoost 模型實作與相關實驗、討論
王皓正：DNN 模型實作、模型間的優劣比較
葉光哲：嘗試各種預處理、Linear Regression 模型實作

## Preprocessing and Feature Engineering

1. Missing Data：以插值處理，取前後項平均，第一筆、最後一筆或前後項有缺漏的資料則直接以前項或後項補。
2. 加入平方項：將訓練和測試資料的平方項也當作 feature。
3. Normalization：訓練和測試資料扣掉平均值後除以標準差。
4. 參考前幾週的天氣：我們認為登革熱疫情受氣候環境影響，而氣候變化有連續性，因此預測時需參考前幾週的天氣變化。
5. 參考前幾週的疫情：登革熱疫情以週為單位來看的話具有連續性，因此可以藉由前幾週的疫情推算上升或下降的幅度，幫助估計這週的疫情。
6. 參考後幾週的天氣：當週疫情與當週天氣有高度正相關，而未來天氣與當週天氣也有高度正相關，因此預期當週疫情與未來數週的天氣也有正相關。
7. 利用 week_of_year 分類：因為氣候原則上每年會規律循環一次，因此不同年的同一週應該有相似的氣候，而氣候又影響登革熱疫情。
8. 去除與疫情較低相關的 feature：將各項 feature 和 total_cases 計算相關係數，並將相關度非常低的 feature 去除。
9. 去除非連續性資料：當使用線性迴歸的時候，單項資料對於疫情預測的結果是線性的；而第幾年、第幾週或是經、緯度坐標對於疫情的結果明顯非線性，因此將這些資料（前 8 個 columns）去除。

我們在 5/28 simple baseline 時間線以前，使用 Linear Regression 模型，配合第 1、2、3、7、9 項的預處理之後，成績為 25.2909，後來再加上第 4 項的預處理之後，成績進步到 23.5361，但在這個模型之下成績一直卡在 23.5 附近，因此萌生了嘗試新模型的想法。

在 6/11 strong baseline 時間線以前，我們改用淺層的 DNN 模型，配合原先的各種預處理，並開始加入 validation，產出大量由切除不同 validation set 的 training set 訓練出的模型，同時預測結果再取平均增加預測穩定度，使成績進步到 22.8822。

在 6/22 分享會過後，受台上同學啟發，我們加入第 5 項預處理技巧，並試著使用 XGBoost，預測的結果開始可以抓到每年夏季疫情的峰值，而這是先前兩個 model 沒有抓到的，最終得到日前的 Leaderboard 成績 19.3605，因此非常感謝同學的不吝分享。

## Model Description

➢ Linear Regression

預期疫情數為各項 feature 的數值乘上特定權重之後，再全部加總的結果，並且使用對 training data 要有 minimum squared error 的方式得到此特定權重，並以此權重預測未知疫情；此外為了降低對於 training data 的 overfitting，還加入了 regularization。

實作上，我們直接使用 sklearn 裡 linear_model 的現成套件，並且因為兩個城市有不同的地理、氣候條件，所以將兩個城市分成兩個 model 分開 train。

1. 參數
   (1) 前處理中的參數：使用過去資料的多少週、未來資料的多少週、過濾相關係數多低的 threshold、是否加入平方項或立方項。
   (2) linear_model.Ridge 函數中的參數：代表 regularization 程度的 alpha、normalize 為 True 或 false

## ➢ Deep Neural Network

再來我們採用的是 DNN 的模型，由於上課時教授提過 NN 較一般 Regression 稍強大之處就在於在 data 較少量時依然可以維持高的 performance。正符合我們這題的需求，由於 training data 的資料數量沒有那麼多，正好可以利用 NN 的特性將資料拉到一個高維空間中以提取更多 feature 的資訊。

NN 架構中，我們前處理中除了第五項外的其他步驟都使用了。和先前一樣將 San Juan 和 Iquitos 城市的資料分別建成兩個不同模型。

1. 參數
   (1) 前處理中的參數：同上 Linear Regression 的前處理參數，這邊直接套用
   (2) NN 模型參數：模型架構、每層的 neuron 數量、dropout 量等等。
2. Validation & patience
   取十分之一的資料作 validation set，20 次 validation loss 無下降則 early stop，由於每次只取了九成資料作 training，故使用了多個(約 20~30 個)train 出來的 models 來取平均

## ➢ XGBoost

觀察訓練資料，可發現 San Juan 波動大，而 Iquitos 差距較不明顯，因此我們針對兩個城市訓練出不同的模型。預處理時使用前述的技巧 1、2、4、5，取前 5 週的氣候資料與登革熱疫情資料（label）。

1. 參數
   只調整三種參數：Objective Function、Evaluation Metric、Boosting Iterations。
   a. Objective Function：reg:linear
   b. Evaluation Metric：MAE
   c. Boosting Iterations：San Juan 的資料大約在第 5～8 個 iteration 收斂，而 Iquitos 在第 3 或 4 個 iteration 即收斂，收斂後開始出現些微 overfitting，但若要能在 tesing set 上成功預測高峰，則必須訓練 15 個 iteration 以上，因此最後選擇訓練 20 個 iteration。

2. Validation
   訓練資料非常少，因此每次訓練時隨機取一成的資料作為 validation set，

另外九成作為 training set，觀察不同 training set 訓練出的模型是否能預測出 peak。最後選擇 random seed = 4。

## Experiments and Discussions

由於我們最後最好的成績是使用 XGBoost，故在實驗與討論上我們將多著墨在 XGBoost 上，最後再整體比較三個方式的優劣。

- **Linear Regression：**

各種資料預處理對於預測準確度的影響（這裡以 public score 作為評量標準）

1. 原始情況：

| 使用的預處理 | public score |
|---|---|
| 去除非連續性資料、normalization | 27.6130 |

2. 針對是否加入資料的平方項與立方項進行實驗：

| 使用的預處理 | public score |
|---|---|
| 加入平方項 | 26.3365 |
| 加入立方項 | 26.5000 |
| 同時加入平方、立方項 | 27.0144 |

實驗發現，加入平方項或立方項可以明顯增加預測準確度，但將平方、立方項同時加入時，預測準確度卻不如只加入平方或立方項，因此最後選擇只加入平方項。

3. 針對是否加入對 week_of_year 這項 feature 進行 to_categorical 進行實驗：

| 使用的預處理 | public score |
|---|---|
| 不加入 | 26.3365 |
| 加入 | 25.2909 |

實驗發現，將「這是否為當年的第 x 週」當作一項 feature，對於疫情預測有顯著效果。探討其主要原因是每年的同一週會因為氣候相似而有相近數目的疫情數。

4. 針對是否要參考前幾週的天氣數據進行實驗：

| 使用的預處理 | public score |
|---|---|
| 加入前 1 週的天氣數據 | 25.0361 |
| 加入前 3 週的天氣數據 | 24.6779 |
| 加入前 5 週的天氣數據 | 23.8269 |
| 加入前 10 週的天氣數據 | 24.1394 |

實驗發現，加入過去的天氣數據對於預測有顯著效果。探討其主要的原因是天氣對於疫情有一定的潛伏期，可能是在某些天氣情況下能使得蚊子大量繁衍，並在一段時間之後才會導致疫情數提高。

5. 針對是否要參考後幾週的天氣數據進行實驗：

| 使用的預處理 | public score |
|---|---|
| 加入後 1 週的天氣數據 | 24.0240 |
| 加入後 3 週的天氣數據 | 24.6779 |
| 加入後 5 週的天氣數據 | 23.7067 |
| 加入後 10 週的天氣數據 | 25.3053 |

實驗發現，加入未來數週的天氣數據對於疫情預測的效果並不顯著。猜測原因是未來數週的天氣數據和當週或是過去數週的天氣數據有高度相依關係，因此重覆加入有類似性質的 feature 並不會對結果產生太大的影響。

6. 針對是否要將與疫情數相關性較低的 feature 過濾進行實驗：

| 使用的預處理 | public score |
|---|---|
| 過濾相關係數小於 0.3 的 feature | 28.1226 |
| 過濾相關係數小於 0.2 的 feature | 28.0793 |
| 過濾相關係數小於 0.1 的 feature | 25.4231 |
| 過濾相關係數小於 0.05 的 feature | 24.8870 |

實驗發現，將 feature 過濾會讓預測的準確度下降。猜測會導致這樣的結果是因為 feature 向量和 total_cases 向量的維度都不算低，所以其實各項 feature 與 total_cases 計算出來的相關係數都相當低，在相關係數都這麼低的情況之下，某項 feature 比另外一項 feature 對於 total_cases 有較高的相關係數並不能代表這項 feature 對於疫情預測較另一項 feature 更有影響。

7. 針對是否使用 regularization 避免 overfitting 進行實驗：

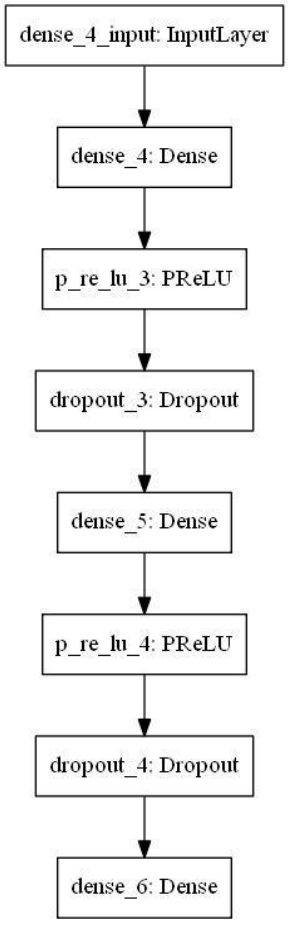
| 使用的預處理 | public score |
|---|---|
| 設參數 alpha = 0 | 23.8269 |
| 設參數 alpha = 0.5 | 23.6298 |
| 設參數 alpha = 1 | 23.5361 |
| 設參數 alpha = 2 | 24.0938 |

實驗發現，加入 regularization 確實能稍微避免 overfitting 的現象，使得預測準確度有些微提高，因此選擇加入 regularization，並設參數 alpha = 1。

預處理總結：經過以上的各項猜測與實驗之後，發現「加入未來天氣數據」及「將低相關的 feature 過濾」並不符合預期，而其它預處理都對疫情預測有顯著效果，因此除了這兩項以外，其它的預處理我們都有使用，並且除了將此處理過後的資料用在 Linear Regression 模型之外，之後的 DNN 及 XGBoost 也都是以此為模板進行些微調整。

- **Deep Neural Network：**

試了多次結果後的 DNN 模型如下：



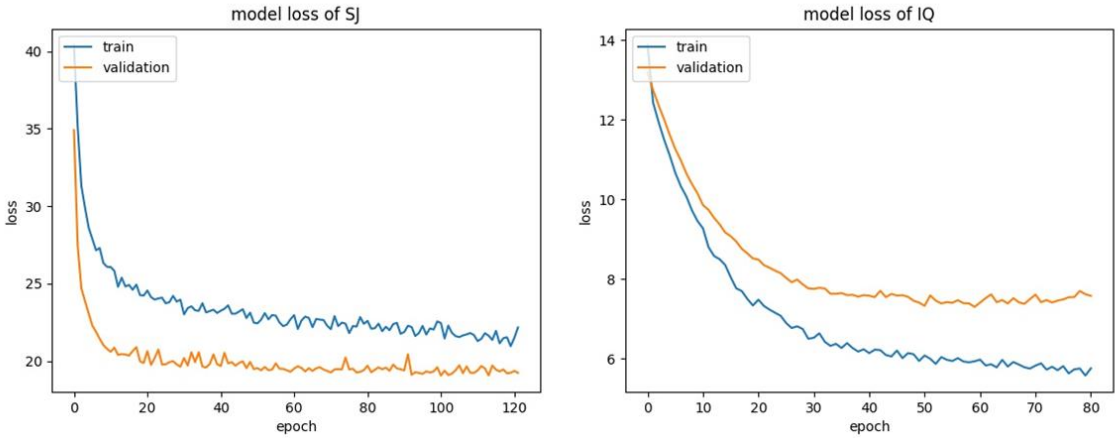

```
def get_model(dim):
    model = Sequential()
    model.add(Dense(32, input_shape=(dim,), kernel_regularizer=regularizers.l1(0.01)))
    model.add(PReLU(alpha_initializer='zero', weights=None))
    model.add(Dropout(0.4))
    model.add(Dense(16,kernel_regularizer=regularizers.l1(0.01)))
    model.add(PReLU(alpha_initializer='zero', weights=None))
    model.add(Dropout(0.4))
    model.add(Dense(1))
    model.compile(loss='mae', optimizer=Adam())
    model.summary()
    return model
```

由於網站上的評分方式是 MAE，所以 loss 的運算方式就與之相同。雖然前者有提到可將 DNN 拉到高維度上去分析較精密的資料，但多次的實驗結果發現在第一個維度上將維度拉上去並沒有好的結果，最後慢慢降下來才得到現在這樣的結果。每個 layer 都有再過 regularization 及 dropout，最後也有做 early stopping，都是為了不讓資料過於 overfitting 在現有少數的 training data 上。

在前面 validation 也有提到過，最後由於只取了 9 成的資料作 training，故我們生成多個 model 後看其 loss 值取出在 validation set 上表現較好的幾個 model 來預測並作平均。

我們最後在 SJ 和 IQ 城市各生成了 491 個 models，篩選後 SJ 取了 33 個 models 而 IQ 取了 22 個 models，不做 cross validation 的原因是怕 training 時沒有取到好的 loss 點成績，這樣還要反覆做多次，不如就直接每次隨機取 0.9 作 training data。

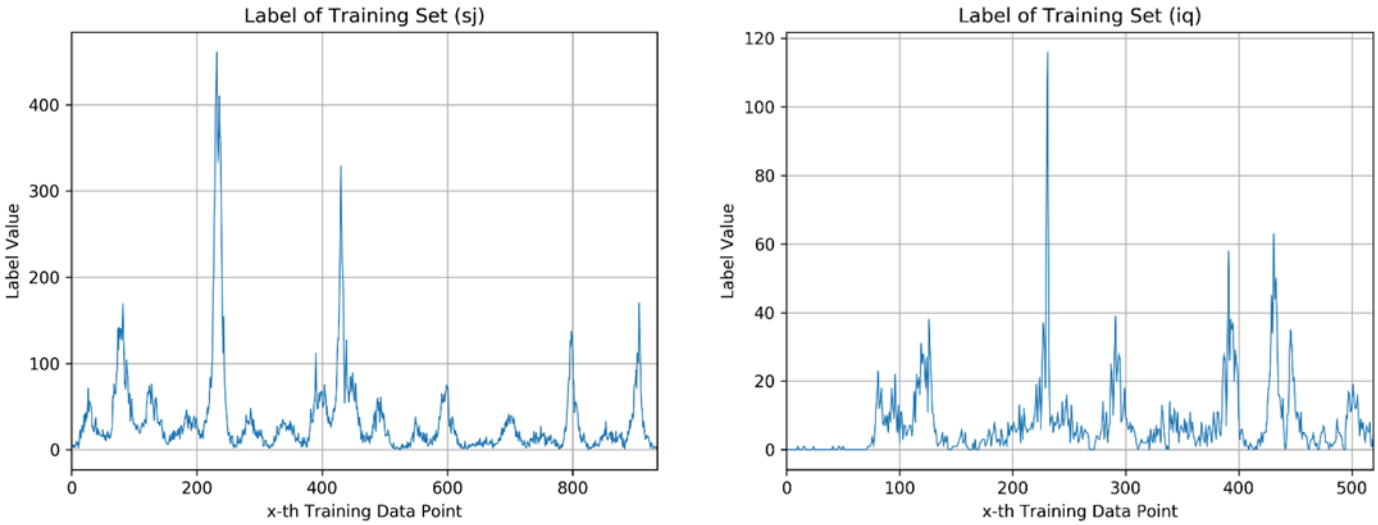
以下是 trainging 時 loss 對時間的關係圖：




在前處理部分也有提到過，由於兩個城市 label 差異很大，所以我們分成兩個 models 來處理，很有趣的是 SJ 和 IQ 的 validation loss 都降到比上繳的成績(22.8822)還要低，尤其是 IQ 城市的，低上超級多，最主要原因就是資料量太少的緣故，造成資料很快地就 overfitting 在 現有的 training set 及 validation set 的關係。而且由於最後成績的計算是兩個城市的 MAE，所以分開來看也有所疑慮。

- XGBoost：

1. 觀察資料及問題——高峰的重要性

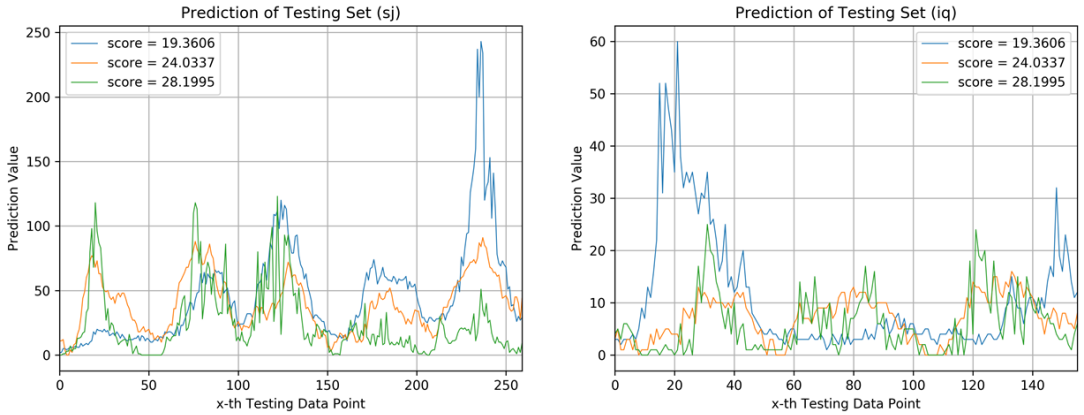



上方圖表分別為 San Juan 及 Iquitos 的 training label 走勢，下方圖表則為三份對 testing set 的預測，三份表現各不相同。

先觀察上方圖表，可發現登革熱大約一年有一次較矮的高峰，推測為夏季氣候造成，而不定時會出現極大的高峰，代表嚴重疫情爆發，這樣的爆發可能達到正常值的十倍以上。

再觀察下方預測結果與 leaderboard 分數的關係，可發現高峰越明顯的預測結果

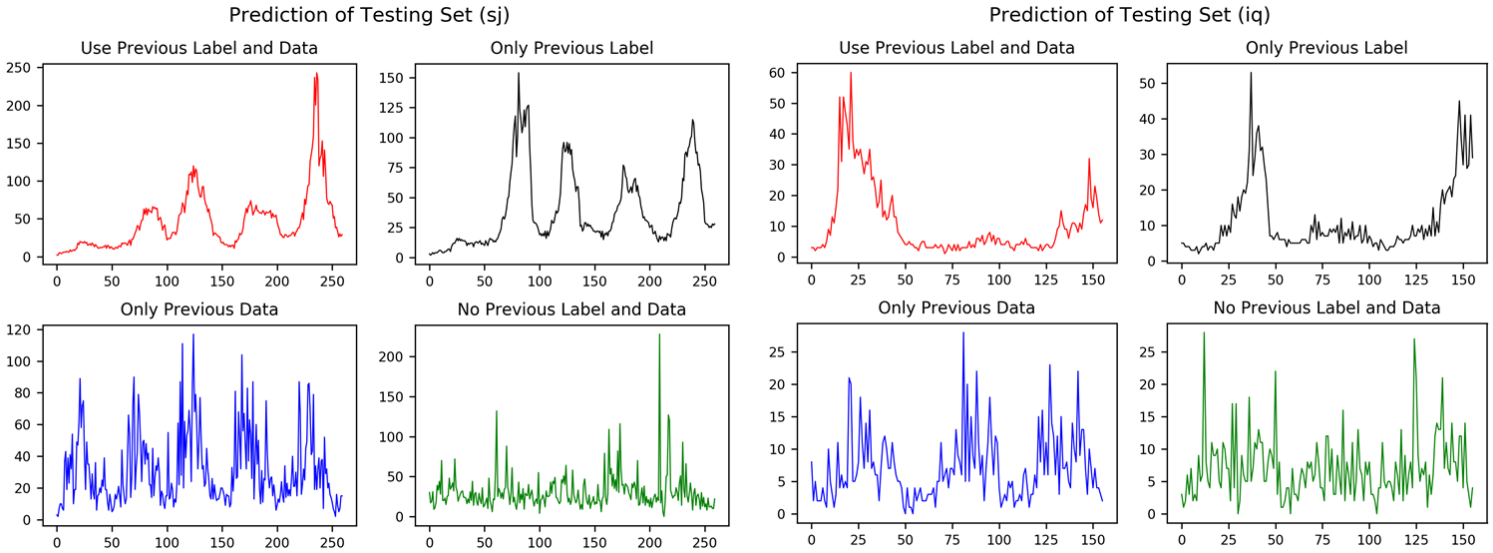
可以拿到越高的分數，因此我們推測，這個比賽的高分關鍵就在於預測高峰，也就是拉開一般預測值與高峰預測值的差距。




2. 歷史資訊對預測高峰的影響

a. 歷史資訊的必要性

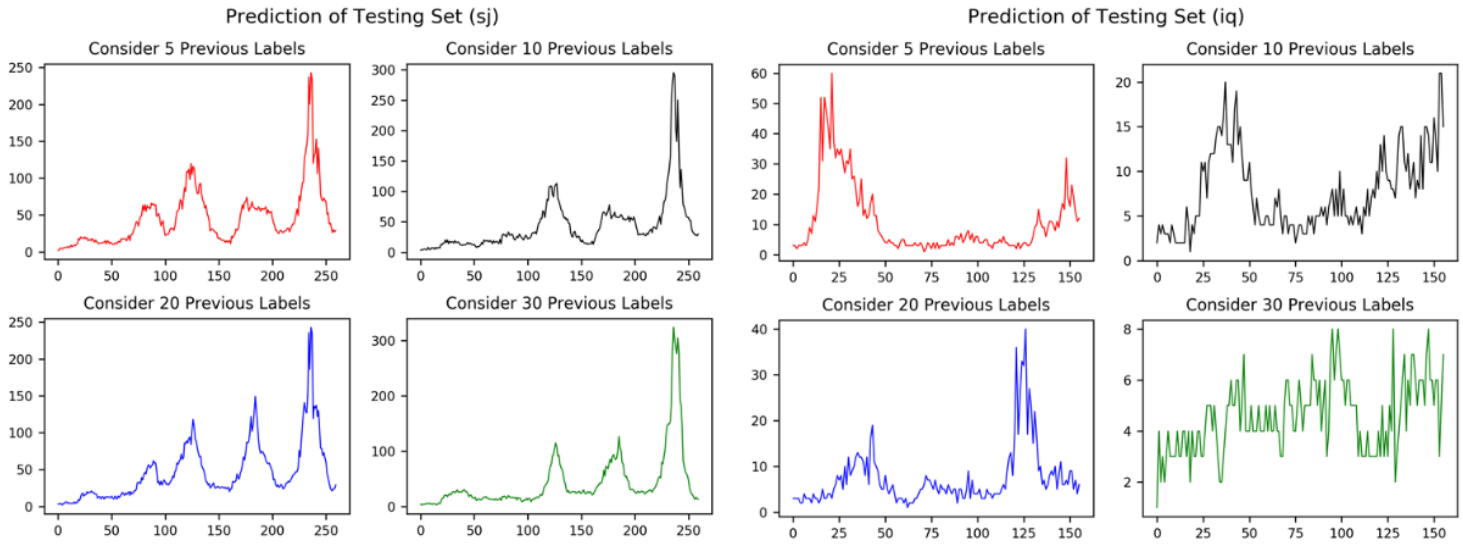
以 XGBoost 模型為例，其使用前五週的氣候資料與疾病資料，若不使用前五週氣候資料或 label，預測結果如下圖：




由上面兩張圖可發現，若不使用歷史資料，預測結果有如雜訊，無跡可循，單純使用歷史氣候資訊可以判定季節性的疾病高峰（夏季登革熱盛行），而單純使用歷史 label 預測出的結果和原始結果最相似。因此我們推測，預測高峰的最大功臣即為歷史 label，而歷史氣候資料則有輔助功能。

b. 歷史 label 的多寡

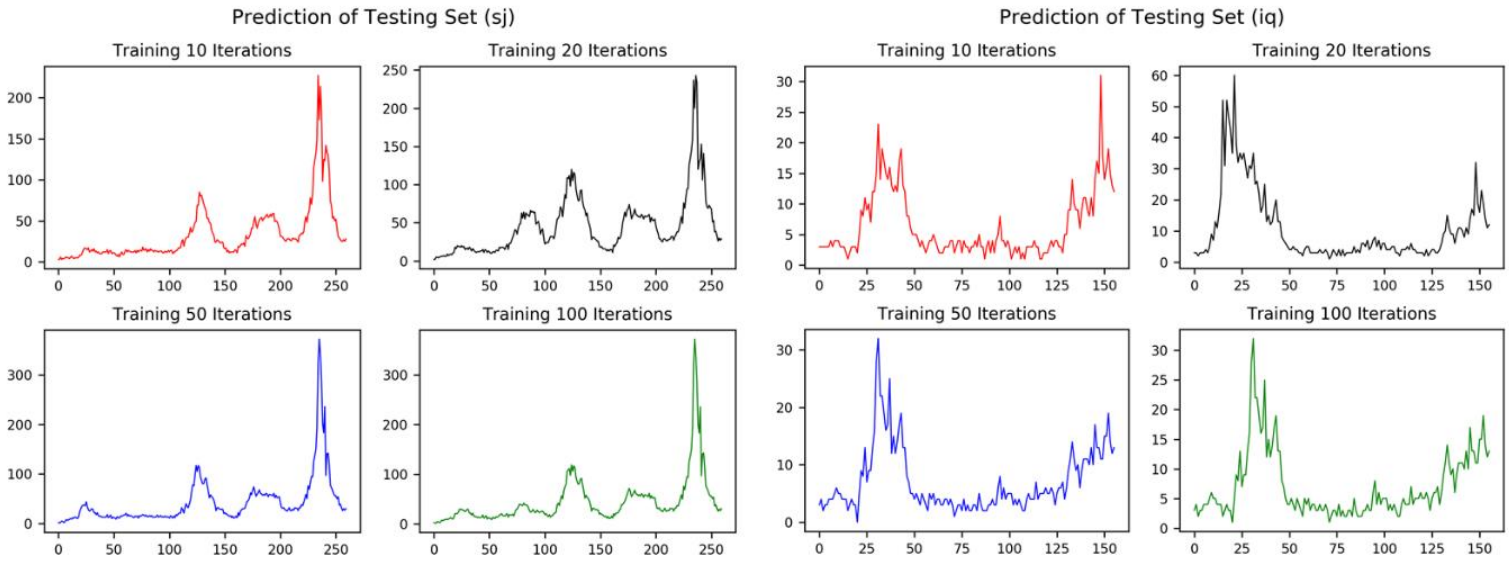
承上點，知道歷史疾病資料對預測精確度的影響後，我們使用前五週的氣候資料與前五、十、二十及三十週的 label 來預測 testing data，結果如下：




由 San Juan 結果可知，使用前五週或前二十週的 label 可預測出相近的四個高峰，而由前十週或前三十週的 label 則只能預測出三個高峰。而由 Iquitos 可發現，使用前十週或前三十週的 label 的預測結果形如雜訊，而由前二十週的 label 預測出的高峰較不明顯。因此，使用前五週的 label 預測出的結果應該是最好的。

3. XGBoost 模型的訓練次數

XGBoost 的訓練時間非常短，且 training 和 validation loss 下降非常快十次以內就會收斂，下表比較不同訓練次數產生之 model 在 testing set 上的表現。

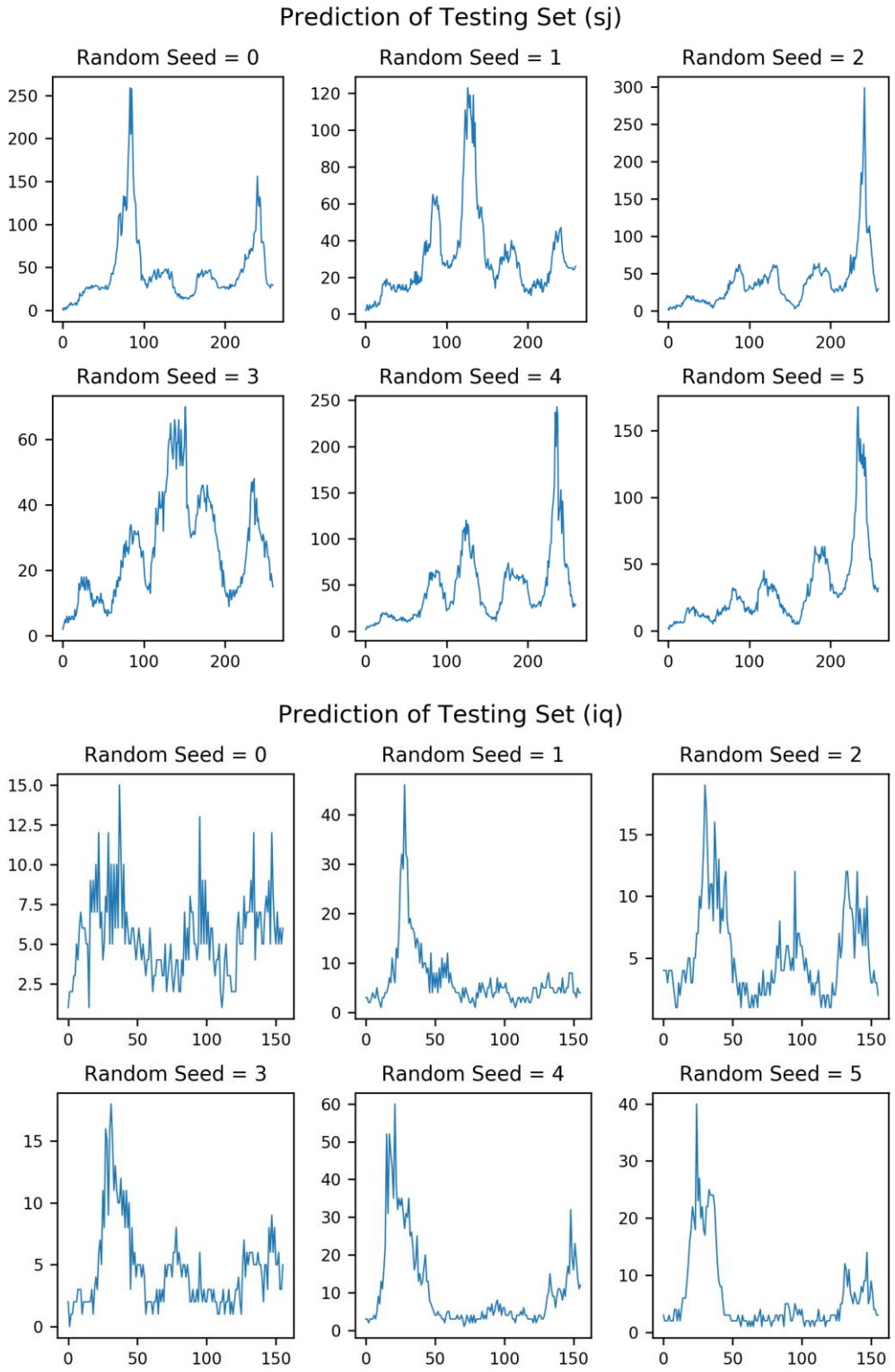



由上表可發現，對 San Juan 的資料來說，訓練 50 或 100 次可增加高峰頂點的值，但訓練 20 次所產生的模型可以在 testing set 上預測出最多高峰。而對 Iquitos 來說，訓練 20 次可以預測出最明顯也最高的高峰。因此，我們最後選擇

訓練 20 次的模型作為最後的模型。

4. 訓練資料過少

整個 training set 只有 1456 筆資料，其中 San Juan 有 936 筆，Iquitos 則只有 520 筆。若每次訓練時取一成的資料作為 validation set，則 training set 又更小了。在 XGBoost 模型上以不同 random seed 得到之 validation set 所訓練的模型在 testing set 上的表現如下：



由上面兩張圖表可知，不同 random seed 所得結果差距非常大，若改變 random seed 則模型其他參數也要跟著調整，才可能得到好的結果。此為訓練資料過少所造成的麻煩，增加往後的訓練難度。最後我們綜合兩城市之高峰極值與高峰數量，選擇 random seed = 4。

## Conclusion on models & our training process

在作業剛開始拿到資料時，由於資料上的數字基本上都是連續性的，所以也沒有想太多就直接先套用 Linear Regression 的方式，在數天內很快地就突破了 simple baseline，原本想說可能再些微地調整參數或預處理的方式能更讓 linear regression 的成績更優秀，但是剛突破 simple baseline 後就卡在瓶頸中了。

所以之後開始嘗試新的 model，也就是第二點所提到的 DNN 模型，DNN 在調整上比較複雜一點，由於架構不如想像中地需要拉到高維度上，所以花了一點時間，最後還是調整到現在的狀態並剛好過了 strong baseline。

但是 DNN 模型架構又有和之前 LR 架構類似的狀況就是遇到瓶頸卡住了，各種方式去 train 都無法得到更好的結果，在 6/22 分享會後才曉得可以利用 Xgboost 的方式搭配我們一開始就已經研究好的 Linear Regression 的 model 去做進一步的改善，成績就來到最後的 19.3605。

有可能是資料真的較少的問題，我們很難從 validation loss 上判斷我們的 model 良好與否，所以只能透過少量的上傳機會來瞭解本次做出來的 model 有效與否。實為可惜，如果可以找到一種方式能在本機端更能判斷 model 的好壞，可能能作出更好的成績。

三個 model 比較下來，其實 Linear Regression 算是中規中矩，多半只能靠 preprocessing 的修改方式使之變好，而 DNN 模型較 LR 多了一點變化，故成績會較 LR 要好，但 LR 配上 Xgboost 後會比 DNN 好的原因，是因為這次的資料沒辦法讓 NN 有太大的發揮空間，因為都是一些離散性的數字，NN 的作用之處是能從 feature 中發現更多的 feature 所在，但這個作用在 NN 的實驗中就已經發現不太管用，所以 NN 模型的方式才沒有辦法有所突破。